Instruction book

ver. 1.01

●ご使用上の ⚠注 意及び ⚠警 告●

：ご使用の前に必ずお読み下さい。

■ SC7000は、SystemCreatorシリーズ MPUボード JES－7001専用のC言語開発環境です。
予めご了承ください。

■ SC7000は、コンパイラ「ANSI－C」に準拠していますが、システムの制限により使用できない関数や機能があります。予めご了承ください。

■ 動作環境 ： RS232Cのシリアル通信ポートを装備、または市販のUSBシリアル変換アダプタ（ケーブル）を接続したWindows互換機。

※全ての Windows 互換機（メーカー製、ショップブランド、自作・改造されたパソコン等）での動作を保障するものではありません。機種によっては正常に動作をしないものがありますのでご注意下さい。
※USBシリアル変換アダプタ（ケーブル）をご使用の場合は、パソコンのOSに対応したアダプタをご使用ください。

■ 対応OS：Windows Vista／XP／2000 ※その他OSについては、サポートいたしません。

■ インストールに必要なハードディスク空き容量：110 Mbyte 以上

■ CD－Rは、落としたり、水にぬらしたり、湿度の高いところに保管したり、熱いものに近づけたり、高温になるところに保管したり、キズをつけたり等不要なストレスを与えないで下さい。
動作不良や破損の原因となります。

■ 製品（パッケージ・ケース・CD－R）の一部には、鋭利な部分や先の尖った部分があります。
ケガや事故の原因となりますので、取り扱いには充分注意して下さい。

■ 本製品をご使用になり得られたあらゆる結果及び効果については、当社はその一切の責任を負いません。

■ 著作権者の許諾なく、本製品の複製、賃貸、その他類する行為は、法律で禁じられています。

■ 製品の仕様及び、価格等については、予告なく変更する場合があります。

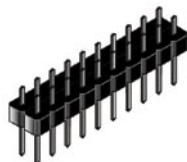
Contents

製品のご案内

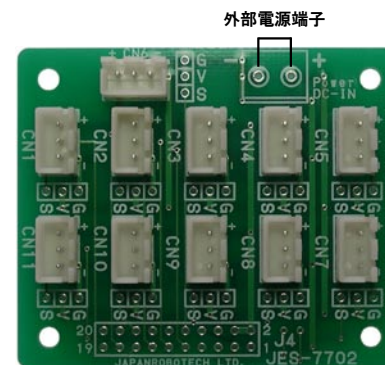
MPU ボード　JES-7001

　C言語プログラミングソフト「SC7000」専用のコントローラボードです。
　信号の入出力は、ソフト上で任意に選択・設計が可能ですので、自作機器のマイコン制御へフレキシブルに対応が可能です。別売のConnectorボード JES－7002を使用することにより、外部機器制御がより容易となります。 ※プログラムの書き込みには、別途「Writingボード JES－7003（プログラミングソフト SC7000 CD－R付）」が必要です。

【仕　様】
●メモリ／ MPU ES7000 標準装備・プログラム領域 8Kbyte・RAM領域 256byte
　　データ格納領域 64Kbit（I2Cメモリ：IC交換により容量増大可）標準装備。
●最大入出力ポート数／11ポート
　※ 全ポート　Digital In / Out　選択・設計可能。
　　内 Analog In 選択・設計可能ポート×4ポート、PWM信号出力 選択・設計可能ポート×2ポート、COMポート 選択・設計可能ポート×1ポート
　※「選択・設計」：ソフトウェア SC7000 上でハードウェアの設計を行います。
●電源電圧／ DC6V～12V
●サイズ／W60×H50mm
●付属品

・ピンヘッダー　2×10P ×1コ
・ネジ付スペーサ　M3×11mm ×4コ
・ビス　M3×6mm ×4コ
・ナット M3×4コ

※付属のパーツは、ボードの固定や「I/O Connector ボード JES－7002」を使用する際のジョイント用に使用します。

I/O Connector ボード　JES-7002

【組立図】

　「MPUボード JES－7001」の入出力端子をポート毎の3Pinコネクタへ変換するボードです。コネクタへ供給される電源は、「MPUボード　JES―7001」の回路電源（＋5V）が供給されます。コネクタ毎に付属する「G・V・S」端子は、「外部電源端子」より供給される電源により外部機器を制御する場合に使用します。＋5V以外の電圧で制御する場合に大変便利です。
※外部電源端子を使用する場合は、注意が必要です。使用を誤ると「MPUボード他、外部機器等を破損する場合があります。

※3Pinコネクタへは、別売の「ケーブル・3Pin　RDP－804または、805」が接続可能です。

【仕　様】
●入出力コネクタ数／11●外部電源端子付
●サイズ／W60×H50mm
●付属品
・ピンヘッダー（G・V・S端子用）　1×40P ×1コ
・ネジ付スペーサ M3×20mm ×4コ
・ビス　M3×6mm ×4コ
・ナット　M3×4コ
・ピンフレーム　2×10P ×1コ
・ネジ式端子台（外部電源端子用）

Writing ボード　JES-7003H/P

【仕　様】

●MPUボード JES－7001専用「書き込みボード」です。

●RS232Cシリアル通信「STRAIGHT」ケーブル専用です。

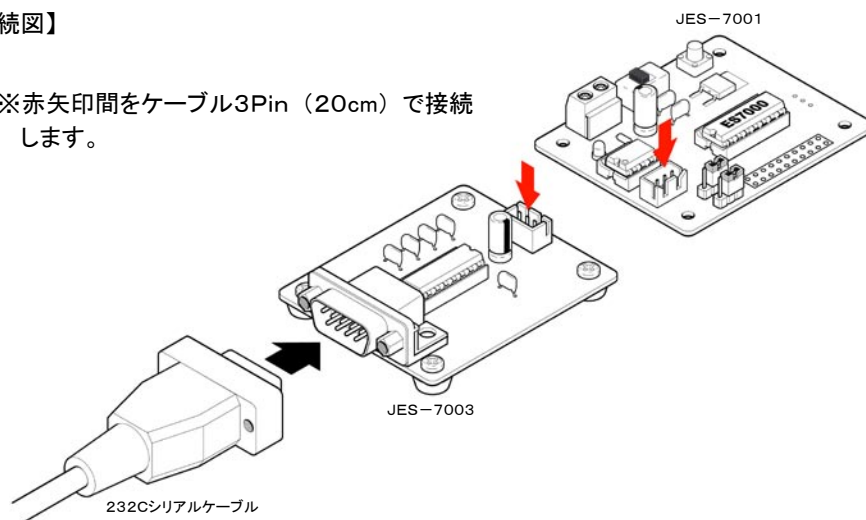
●本製品には、「C言語プログラミング開発環境 SC7000（ソフトCD－R）」が付属しています。

●232C シリアル通信コネクタには、「ピン型（JES－7003P）」と「ホール型（JES－7003H）」を準備しています。ご使用の「232Cシリアルケーブル」のコネクタ形状（ケーブルが「ピン型」の場合「JES－7003H」、「ホール型」の場合「JES－7003P」）に合わせお求めいただけます。

●サイズ／W60×H50mm（232Cコネクタ突起部含まず。）

●付属品　・ケーブル3Pin（20cm）×1本（MPUボードとの通信用）
　　　　　・ゴム足付（基板固定済み）

【接続図】

※赤矢印間をケーブル3Pin（20cm）で接続します。

その他関連品のご紹介

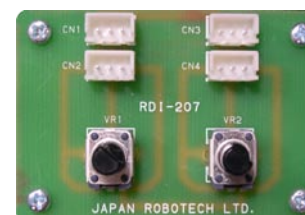
4bit ディジタルスイッチボード　RDI-205A

スライドスイッチを使用したディジタル入力実験用のボードです。最大4ビットまでの接続が可能です。

【仕　様】

●スライドスイッチ使用により、「High／Low」の状態保持が可能。

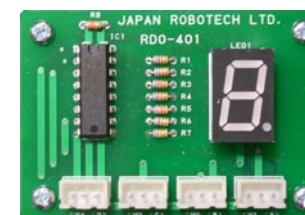
●4ビットまでの接続が可能。

アナログ実験ボード　RDI-207

アナログ入力実験用のボードです。
ボリュームにより0V～5Vまでの出力電圧可変が可能です。

【仕　様】

●ボリュームによりDC0V～5Vまで出力電圧可変可。

●出力2系統接続可。
　※各系統パラレルコネクタ標準装備。

7セグメント LED 表示ボード　RDO-401

ディジタル出力実験用のボードです。4ビットのディジタル入力コネクタに対応した表示を7セグメントLEDへ表示します。

※4ビットの入力（0～15）に対応したパターンを表示します。

※基板上のコネクタ（A・B・C・D）は「CN1（A）」が LSB（Least Significant Bit 最下位）側、「CN4(D)」がMSB（Most Significant Bit 最上位）側です。

その他関連品のご案内

タッチセンサ RDI-201

　接触式センサの仕組が解りやすく、バネの取り付け位置により感度調整も可能です。

●接触部／ステンレスバネ（線径0.3mm f40×50mm）
●サイズ／30×25×15mm（突起部含まず）
●入数／2セット

※要組立 ※ケーブル／RDP－804及びRDP－805別売

アナログ赤外線センサ RDI-202

　アナログ出力。アクティブ／パッシブ双方使用可。赤外線LED用パターン及び、アクティブ／パッシブ切替スイッチ用パターン付。

●赤外線LED×2コ付
●サイズ／30×20×10mm（突起部含まず）
●入数／2セット

※ケーブル／RDP－804及びRDP－805別売

コンパスセンサ RDI-208

●電源電圧／DC 5V
●信号出力2系統：アナログ／デジタル出力の選択、または同時使用可（使用については注意が必要です。）
●方 位：北･北東･東･南東･南･南西･西･北西の8方位、モニターLED1～3の点･消灯パターンによる方位表示。
●デジタル信号出力時：LED4の点灯
●モニターLED ON／OFFスイッチ付（競技会によっては発光を制限される場合があります。）
●サイズ／65×45×15mm（突起部含まず）
※ケーブル／RDP－804及びRDP－805別売

測距センサ RDI-209

　赤外線を照射し、その反射量（距離）に応じたアナログ電圧を出力するセンサです。
　測定可能範囲は、10cm～80cm。 データロガー機能を併用することで、各種実験や様々な機器及びロボットの制御学習が可能です。
　本製品は、「距離に応じた電圧を出力するもの」で、正確な距離を示すものではありません。予めご了承下さい。
●取り付け穴径φ3.2mm
●3ピンケーブル（30cm）付
●サイズ／44.5×13.5×19mm（突起部含む）
●入数／1

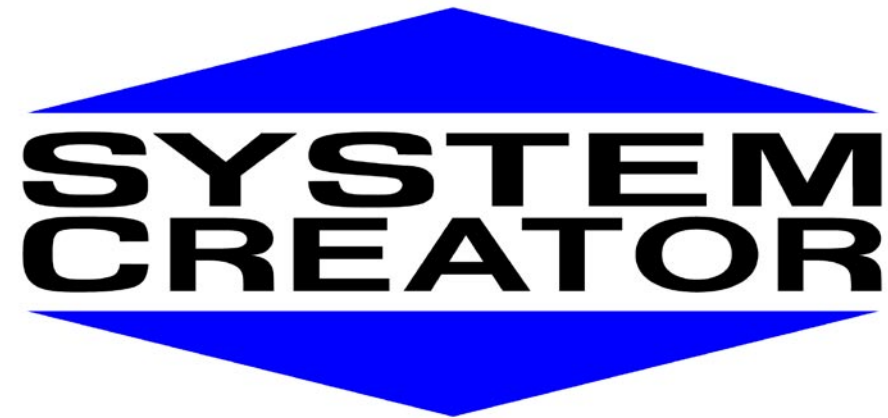
ケーブル・3P（20cm） RDP-804

●コネクタ付3ピン並行ケーブル
●長さ／200mm ●入数／3本

ケーブル・3P（30cm） RDP-805

●コネクタ付3ピン並行ケーブル
●長さ／300mm ●入数／2本

232Cシリアルケーブル RDP-821

●コネクタ形状／ホールタイプ
●長さ／2m
●入数／1本

SC7000 取扱説明書

1. SC7000 操作方法

1-1. 初期起動時の設定

SC7000 インストール後、最初の立ち上げ時、図1 - 2「Pleference Setting」ダイアログで各項目の初期設定を行います。

■ ツールバー「File」(図1-1) →「Preference」
から「Preference Setting」ダイアログが表示され、各項目を選択設定します。

「Preference Setting」は、ご使用のパソコンのプログラミング・データの保存先やプログラム転送時の通信ポートの設定等のプログラミング環境をユーザー専用に設定することが可能です。設定後は、設定環境が優先的に機能し、表示されます。

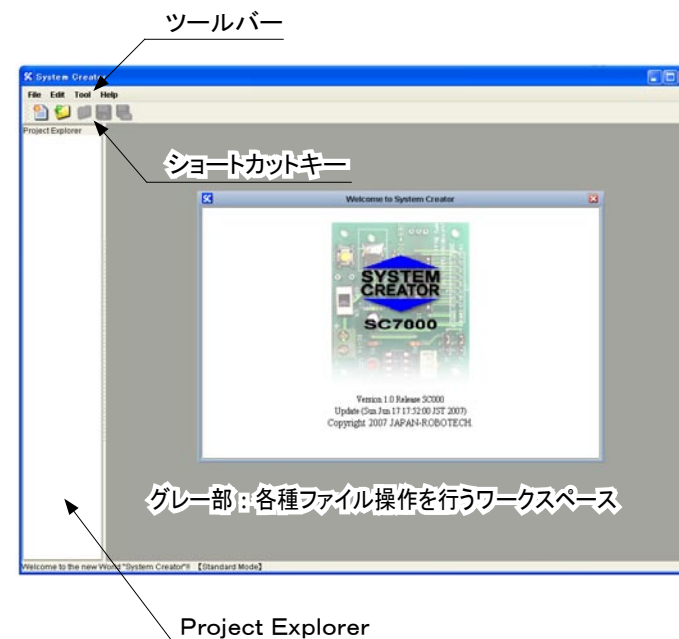
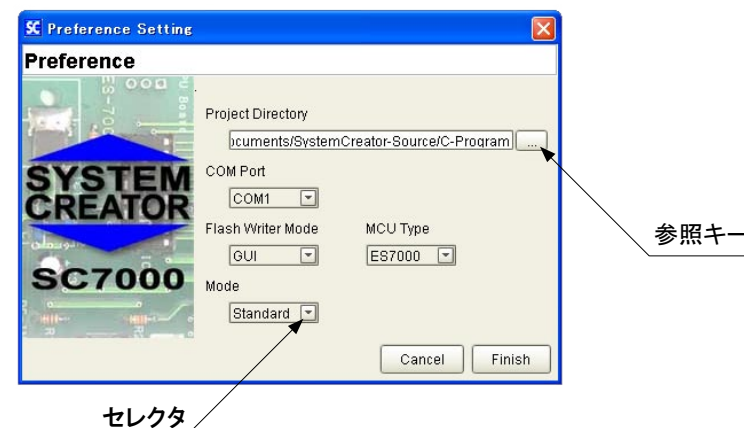
「Preference Setting」で設定された内容は、必要に応じて変更が可能です。設定内容を変更する場合は、

1-1-1.「Preference Setting」設定内容 及び 設定

優先設定可能な内容は、以下内容です。

1) [Project Directory] : 作成したプログラムデータの保存先フォルダの設定
2) [COM Port] : プログラム転送時の通信ポートの設定 (重要)
3) [Flash Writer Mode] : プログラム転送 (ダウンロード) モードの設定
4) [Mode] : プログラム開発レベルの設定
5) [MPU Type] : MPU (マイコン) タイプの選択 (固定)
※SC7000では、MPU ES7000 のみです。

1) Project Directory : 保存先フォルダ設定　図1 - 3

作成したプロジェクト(SC7000開発環境で作成されたプログラム等の総称) の保存先を設定します。
作成したプロジェクトを保存する際や保存したプロジェクトを「開く」際に設定されたフォルダが優先表示されます。

■ 保存先フォルダの設定は、プロジェクト（プログラム）データの管理と作業の簡素化のために行います。

■ 保存されるプロジェクト（プログラム）データは、任意のプロジェクト名の付いた「.xml ファイル」と「その他のファイルフォルダ」（図1－4）の2つ作成されます。
そのどちらか一方でも無い場合は、プロジェクト（プログラム）データとして機能しません。

■ SC7000 インストールと同時に､「マイドキュメント」内へ「保存先フォルダ：SystemCreator－Source / C－Program」が作成され、Project Directory へそのフォルダ名が表示されます。このフォルダを保存先とする場合は、そのまま［Finish］をクリックします。
その他任意のフォルダを設定する際は、［参照］キーから任意のフォルダへ変更・設定します。

（1）保存先フォルダの変更

保存先フォルダを変更したい場合は、以下の手順で行います。

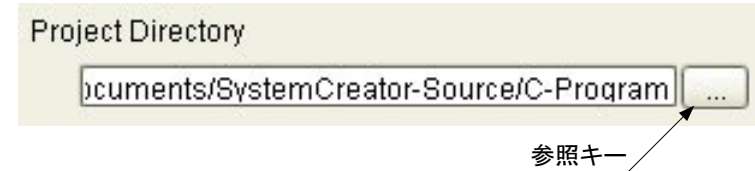
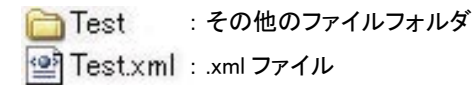
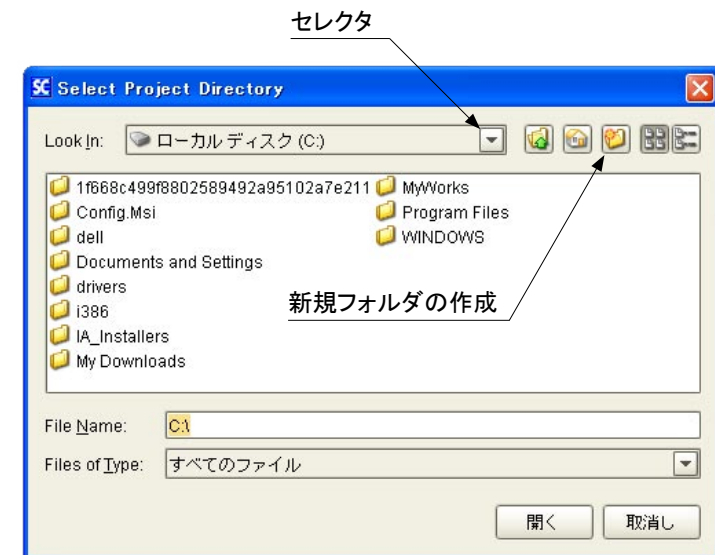
① ［参照］キー ［…］ をクリックします。図1 - 2, 1－3

② ［Select Project Directory］ダイアログが表示されます。図1 - 5

③ **セレクタ**で保存先のフォルダまたは、デスクトップやマイドキュメント、ローカルディスク（C:、D: 等）を指定します。

④ 既成のフォルダを指定するか、新規フォルダを作成し、任意名を付けて、［開く］をクリックします。

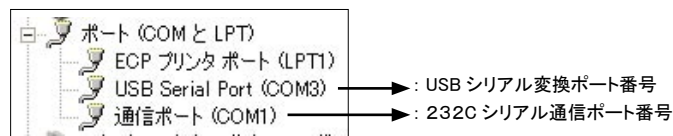
新規フォルダの作成ショートカットキー

…以上で保存先フォルダの設定完了です。

2）COM Port：シリアル通信ポートの設定

232Cシリアル通信に使用しているパソコンのポートを指定します。
プログラムをダウンロードする際やデータロガーを使用する際は、シリアルケーブルを介してデータの送受を行いますので、［COM Port］の設定は必ず行って下さい。

（1）ご使用のパソコンに232Cシリアル端子（9ピン）が装備されている場合は、使用するCOMポート番号に設定します。
※詳細は「デバイスマネージャー」ポート（COMとLPT）で確認して下さい。
※図1 - 6 のセレクタをクリックすると設定可能なCOMポート番号が表示され、COMポートを任意に選択設定します。

（2）USBシリアル変換アダプタ（ケーブル）を使用する場合は、パソコンが認識しているCOMポート番号に設定します。
※詳細は「デバイスマネージャー」ポート（COMとLPT） で確認して下さい。
※図1 - 6のセレクタをクリックすると設定可能なCOMポート番号が表示され、COMポートを任意に選択設定します。

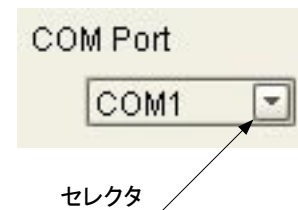
■デバイスマネージャー　ポート（COMとLPT）

［COMポート番号の確認］は、右記の手順で確認して下さい。図1 - 7

3）Flash Writer Mode：ダウンロードモードの設定

図1 - 8のセレクタをクリックし、選択設定します。

通常［GUI］モードで使用します。

※ パソコンの機種やシステム、使用ソフトウェアによっては、稀に通常の［GUI］モードでの正常なダウンロードができない場合があります。

■図1 - 6 ［Select Project Directory］ダイアログ

■1 - 7 COMポートの確認方法

□ Windows XP の場合

□ 2000 の場合
同様のキーワードで「デバイスマネージャー」の「ポート（COM と LPT）」で確認して下さい。

□ Vista の場合
Vista スタートメニュー
→コントロールパネル
→ハードウェアとサウンド
→デバイスマネージャー
の手順で確認して下さい。

●⚠注 意●

● ［COMポート］の設定は必ず行って下さい。
●USBシリアル変換アダプタをご使用の場合は、パソコンが認識しているCOM番号を確認して下さい。
また、USBシリアル変換アダプタは、SC7000起動の前に、 必ずパソコンのUSBポートへ接続した状態で起動して下さい。

この場合、[Console]モードをお試し下さい。

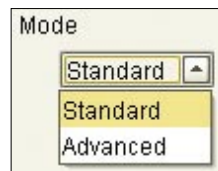
●⚠注 意●

- ダウンロードが正常に行われない原因は様々です。
  まずは、P24「1－5－1. プログラムの転送手順」通りに行っている
  かを確認して下さい。

4）Mode：学習・開発レベルの設定

開発環境レベルの設定が可能です。

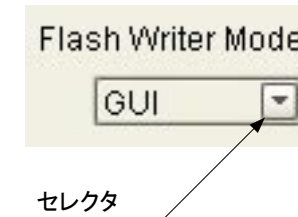
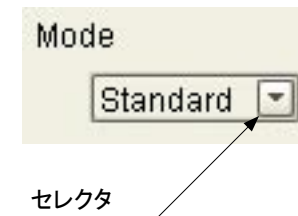
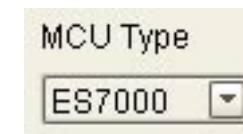
【Standard】：基本的なプログラミング環境です。
図1 - 9

【Advanced】：高度な処理を作成するための環境
です。割り込みや・マイコン動作
全般の高度な知識が要求されます。

セレクタをクリックし、任意に選択設定します。

5）MPU Type：ES7000

C言語プログラム開発環境 SC7000 は、MPU ES7000 専用の開発ソフトですので、MPU Type は ES7000 となっています。図1－10

[Preference」の設定が終了したら、「Finish」をクリックします。

…以上で設定完了です。

1 - 2. プロジェクトの開き方

SystemCreatorでは、プログラミングを行う環境を総称して、「プロジェクト」と称します。プロジェクトの内容は以下の通りです。

① HW（ハードウェア・コンフィギュレータ：ハードウェア設計）ファイル
② main . c他エディタファイル画面
③ Reporter（取得データの書き込み、メモ等に使用できるテキスト入力画面）

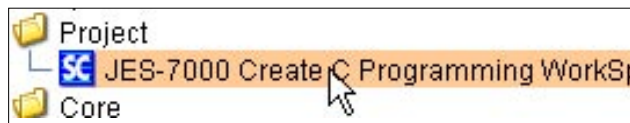
1-2-1. 新規プロジェクトの開き方手順

（1）ツールバー［File］内の［New Project］または、

ショートカットキー をクリックします。図1 - 11

（2）［New Project / File Wizard【～】］が表示されます。図1- 12
※～は、「Preference」の「Mode」設定で選択したモード名が表示されます。

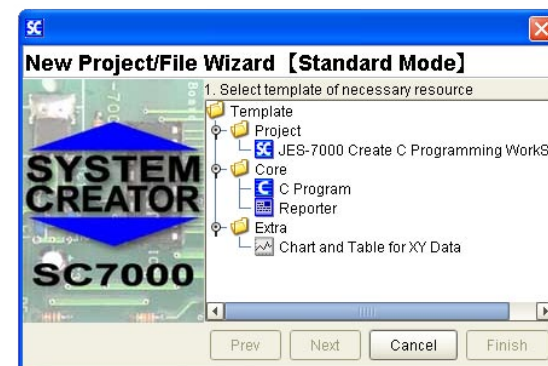
（3）フォルダ［Project］の［JES－7000 Create C Programming WorkSpace］図1 - 13 を選択し、［Next］をクリックします。

■図1 - 13

（4）［Resource Name］の欄に、これから作成するプログラムのプロジェクト名を入力し、［Finish］をクリックします。
※［Resource Name］のテキスト入力は、必ず半角英（大小）数文字のみで行って下さい。

（5）「新規プロジェクト HW 画面」が表示されます。

…以上で新規プロジェクト作成完了です。

■図1 - 11 ［Preference Setting］完了後初期ウインドウ各部名称

■図1 - 12 ［New Project / File Wizard］

1-2-2. プロジェクト ツールバー解説

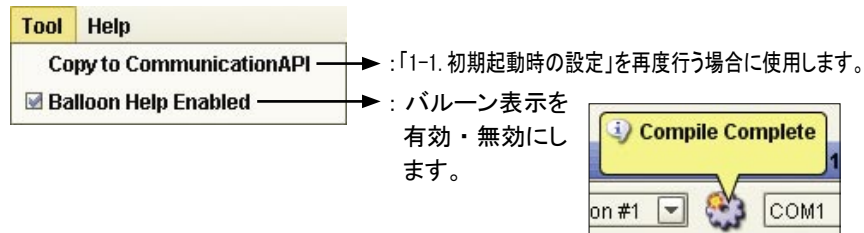
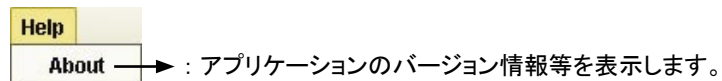
■図1-14

1) File

2) Edit

※ ワープロ同様の編集機能です。
main．c等のエディタファイルの編集時にも使用可能です。

■図1－14 ツールバー と ショートカットキー

3）Tool

4）Help

1-2-3. ショートカットキー解説

: New（新規プロジェクトを作成する）

: Open（保存されたプロジェクトを［開く］）

: Close（アクティブなプロジェクトを［閉じる］）※

: Save（アクティブなプロジェクトを［保存］する）

: Save as ...（アクティブなプロジェクトを［名前を付けて保存］する）

※ 各ショートカットキーをクリックすると関連するダイアログやウイザードを表示します。

※「アクティブなプロジェクト」: プロジェクトエクスプローラ上で選択されたプロジェクトや、ワークスペース上で実際に編集中（カーソル等が動作している）のプロジェクトを指します。

1-2-4. プロジェクトエクスプローラ表示内容

各プロジェクトファイルのウインドウ切替や Close（「閉じる」）機能やプロジェクト内へ各種ファイルの追加処理等を行います。

［Mode：Standard］　図1 - 15

●～－HW：ハードウェア（回路）設計ファイル・ウインドウ

●main . c：プログラミング（プログラムの記述）ファイル・ウインドウ

●Initialize . c：ハードウェア設計反映ファイル・ウインドウ

●ES7000Lib . h：関数ライブラリファイル・ウインドウ

●～－Reporter：リポータ（テキスト入力）ファイル・ウインドウ

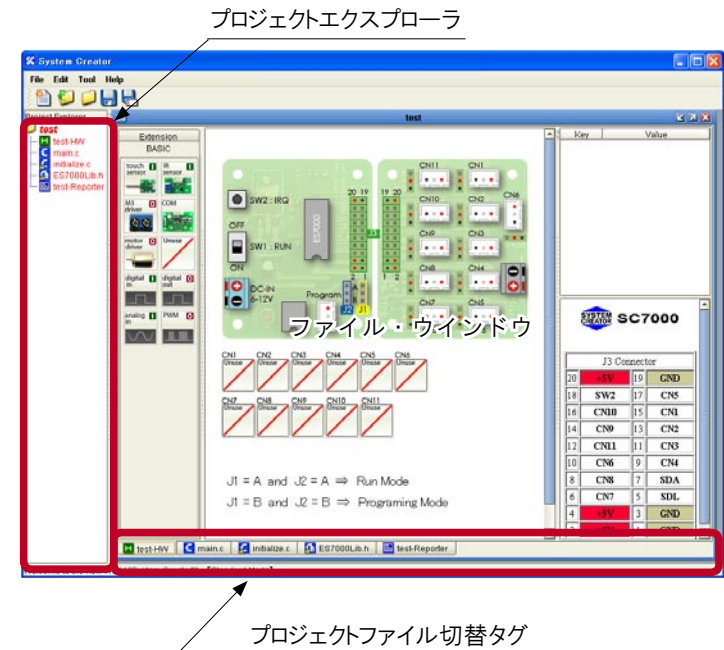
※「～」は、プロジェクト名が反映されます。

※表示される文字の色により、未保存・既保存データを表します。図1－16

※プロジェクトファイル切替タグの表示も同じ表示です。図1 - 17
※各プロジェクトファイルの詳細については、P42「プロジェクト解説」を参照して下さい。

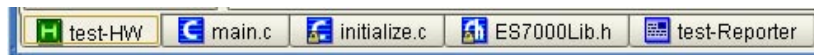
■図1 - 17　プロジェクトファイル切替タグ

※ プロジェクトファイル切替タグには Close 機能やファイルの追加機能はありません。

■図1 - 15　新規プロジェクト HW 画面

■図1 - 16 ［Project Exproler］：Srandard

【編集時の表示：赤文字】

【保存処理済の表示：黒文字】

※ プログラム作成中（編集中）のプロジェクトはプロジェクト名以下「赤文字」で表示されます。プログラムが既に保存されているかどうかの目安となります。

1-3. プログラミングの手順

1-3-1. ハードウェアの設計　H ～HW（「～」はプロジェクト名を表示）

■ハードウェア設計ファイル・ウインドウ（ハードウエアコンフィギュレータ）
：　MPUボードの回路設計を行います。各ポートは、Digital Out / In、Analog In 等設計者の意図により設計可能です。　図1-18

■タイルパレット
：　各種入出力設計用のタイルが格納されています。

■プロパティ
：　ポートの設計可能な内容が表示されます。各項目を選択することで、回路設計が可能です。

1）設計の仕方

（1）タイルパレットの各種機器を表すタイルを図1-19の赤枠内各ポートのタイル配置用のマスへドラッグ＆ドロップで配置し、回路を設計します。

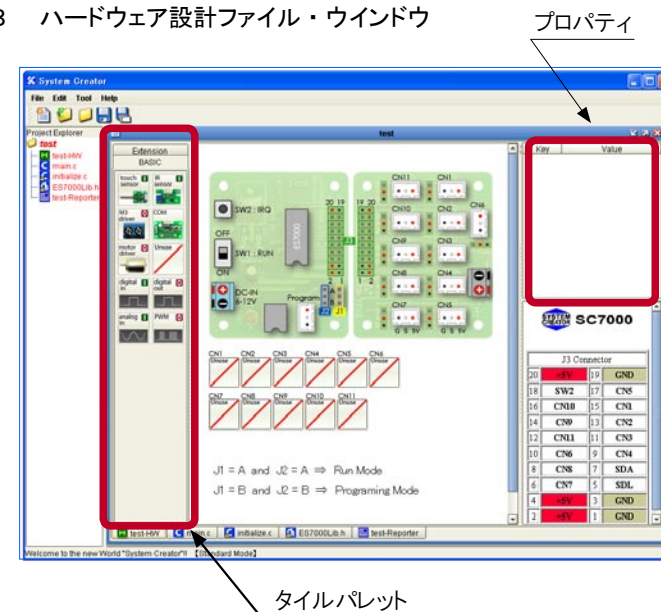
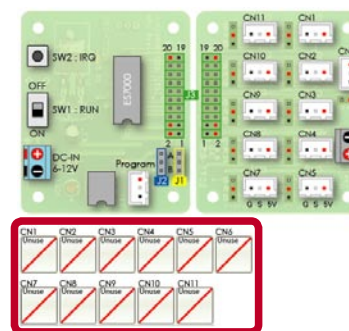
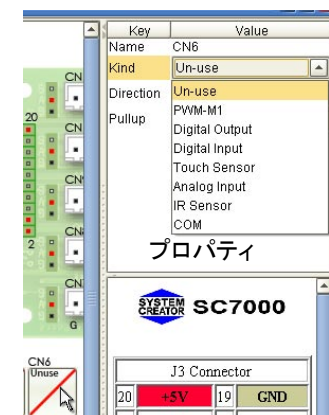
（2）または、赤枠内の設計したいポートのタイル配置用のマスをクリックすると、そのポートのプロパティ（「内容」の意　図1-20）が表示されます。
プロパティ［Kind］のセレクタで「使用したい機器」を選択します。
選択された機器は、赤枠内の指定したマスへタイルが表示されます。
…以上でハードウェア設計完了です。

※ハードウェアコンフィギュレータで設計された内容は、ソースプログラム「Initialize . c」ファイルへ自動的に反映されます。

※タイルについては、P45「タイル解説」を参照して下さい。

●⚠注 意●

※ハードウェアの設計は、ハードウェアコンフィギュレータでのみ行えます。
［Initialize . c］ ファイル・ウインドウでの設計変更はできません。

■図1-18　ハードウェア設計ファイル・ウインドウ

■図1-19　ハードウェアの設計

※　マスをクリックすると、プロパティにそのポートの各種設定内容が表示されます。
「Kind」のセレクタをクリックし、使用したい機器を選択し、設計します。

■図1-20　プロパティ

1-3-2. プログラムの作成（main.c）

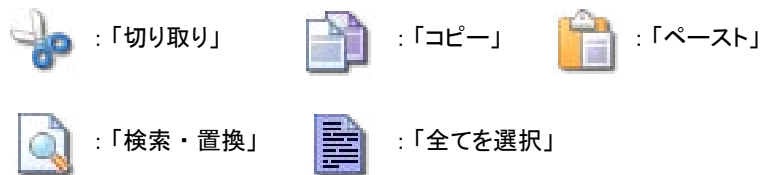
main.c（プログラムの記述）　　図1-21

変数宣言、自作関数宣言、定義、mainプログラムを記述します。mainプログラムは、「ES7000Lib.h」で定義された関数を使用して記述します。

■図1-21 ［main.c］ファイル・ウインドウ

1）ショートカットキー解説　　図1-21

：「切り取り」　：「コピー」　：「ペースト」

：「検索・置換」　：「全てを選択」

※上記各ショートカットキーは、ワープロ　編集と同様に使用します。

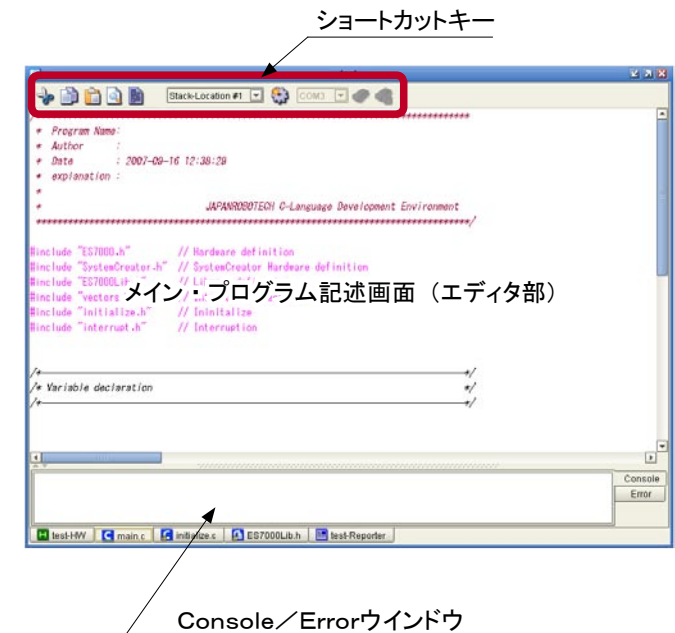
：「コンパイル」　：「転送」

：「転送中」　：「転送中止」

※プログラムを機械語へ変換、転送する際に使用します。

2）キーボードによる操作】

●Ctrl ＋ C：Copy　コピー
●Ctrl ＋ V：Past　ペースト
●Ctrl ＋ Z：Undo　タイピングの取り消し
●Ctrl ＋ Y：Redo「取り消し」を戻す

※［Reporter］ファイル・ウインドウのテキスト入力時にも使用可能です。

3）Stack - Locationセレクタ解説　　図1-22

MPU　ES7000には256byteのRAMが搭載されています。この256byteのRAMは、Stack－Location＃1（初期状態）では、変数やデータの領域として128byte、スタック領域として128byteに分割されています。

プログラムの内容や使用する命令・サブルーチンや割り込み等さまざまな要因により、変数・データ領域の容量アップが必要となる場合があります。

Stack－Locationの選択では別表（図1-23）のように＃1～＃6の6種類の選択ができるようになっています。セレクタ（図1-22）をクリックすると、＃1～＃6が表示されます。コンパイル前に適時選択してご使用下さい。

■図1-22［Stack－Location］セレクタ

■図1-23［Stack－Location］の各領域サイズ

単位：byte

<table>
<tr><th>RAM 容量</th><th>#1</th><th>#2</th><th>#3</th><th>#4</th><th>#5</th><th>#6</th></tr>
<tr><td>241～256</td><td rowspan="8">スタック領域</td><td rowspan="7">スタック領域</td><td rowspan="6">スタック領域</td><td rowspan="5">スタック領域</td><td rowspan="4">スタック領域</td><td rowspan="3">スタック領域</td></tr>
<tr><td>225～240</td></tr>
<tr><td>209～224</td></tr>
<tr><td>193～208</td><td rowspan="6">変数・データ領域</td></tr>
<tr><td>177～192</td><td rowspan="5">変数・データ領域</td></tr>
<tr><td>161～176</td><td rowspan="4">変数・データ領域</td></tr>
<tr><td>145～160</td><td rowspan="3">変数・データ領域</td></tr>
<tr><td>129～144</td><td rowspan="2">変数・データ領域</td></tr>
<tr><td>1～128</td><td>変数・データ領域</td></tr>
</table>

4）「main.c」プログラム記述（エディタ）部解説

```
/* *****************************************************************
 *  Program Name:
 *  Author      :
 *  Date        : 2006-03-04 21:05:01
 *  explanation :
 *
 *****************************************************************/
#include "ES7000.h"          // Hardware definition
#include "SystemCreator.h"   // SystemCreator Hardware definition
#include "ES7000Lib.h"       // Library definition
#include "vectors.h"         // Interruption vector definition
#include "initialize.h"      // Ininitalize
#include "interrupt.h"       // Interruption
/*-----------------------------------------------------------------*/
/* Variable declaration                                             */
/*-----------------------------------------------------------------*/

/*-----------------------------------------------------------------*/
/* Prototype declaration                                            */
/*-----------------------------------------------------------------*/

/*-----------------------------------------------------------------*/
/*  main                                                            */
/*-----------------------------------------------------------------*/
void main(void) {

   init();      //Initialization


   while(1){

      COPCTL = 0;
   }
}
```

インクルード文： プログラムに必要な各種ヘッダーファイルを呼び出します。

グローバル変数宣言 ： プログラムで使用する変数を記述します。

関数プロトタイプ宣言 ： 自作関数を記述します。

main関数定義： 制御プログラムを記述します。自作関数定義等も行います。ES7000Lib．h（関数ライブラリ）で定義された関数を使用して記述します。

1-3-3. プロジェクト保存の仕方

プロジェクトの作成途中や作成後は、出来るだけ保存するよう心がけましょう。

作成したプロジェクトや作成途中のプロジェクトを保存する場合は、以下の手順で行います。

1）プロジェクト名を変更せずに保存する

（1）ツールバー［File］→［Project Save（保存）］の順にクリックします。

または、ショートカットキー 図1-24 をクリックします。

（2）［Preference］で設定した保存先フォルダへ自動的に保存されます。

2）プロジェクト名を変更して保存する

（1）ツールバー［File］→［Project Save As ...（名前をつけて保存）］の順にクリックします。

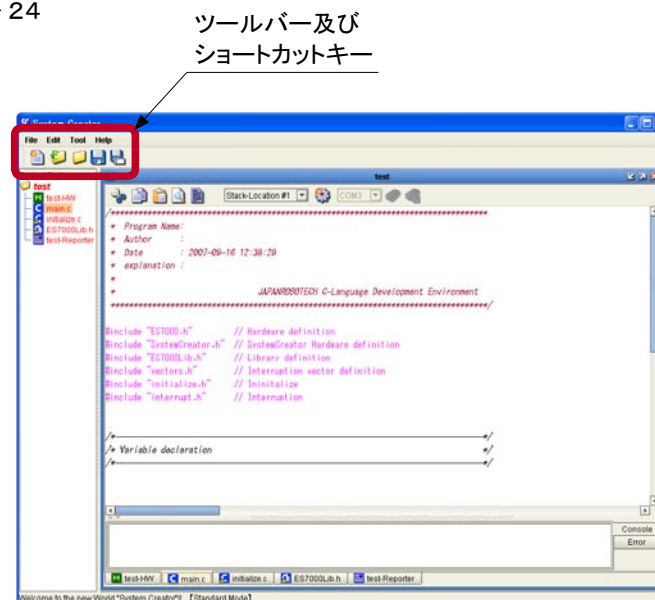
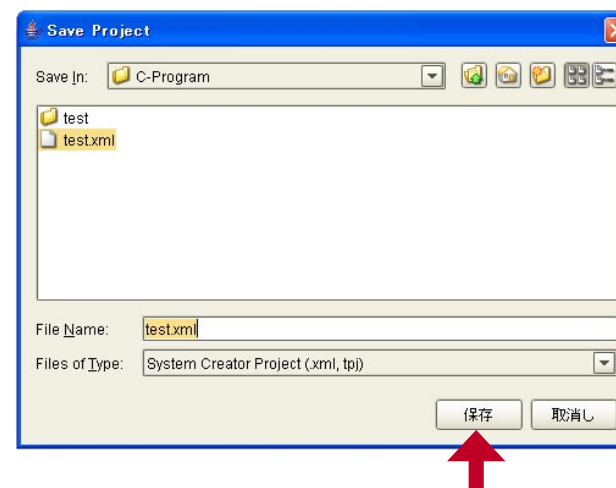
または、ショートカットキー 図1-24 をクリックします。

（2）［Save Project］ダイアログ（図1-25）が表示されます。
※［Preference］で設定した保存先フォルダが優先表示されます。

保存先を変更する場合は、保存先フォルダを変更し、任意のプロジェクト名を入力し、［保存］をクリックします。

※［Project Explorer］で表示される文字が「黒」へ変わります。

…以上で保存完了です。

■図1-24

■図1-25［Save Project］ダイアログ

1-4. コンパイル

プログラムを機械語へ変換します。
C言語プログラムをマイクロコントローラへ転送する前に必ず行います。

コンパイルは、以下の手順で行います。

1-4-1. コンパイルの手順

（1）一連の作業でプログラムの作成が終了したら、

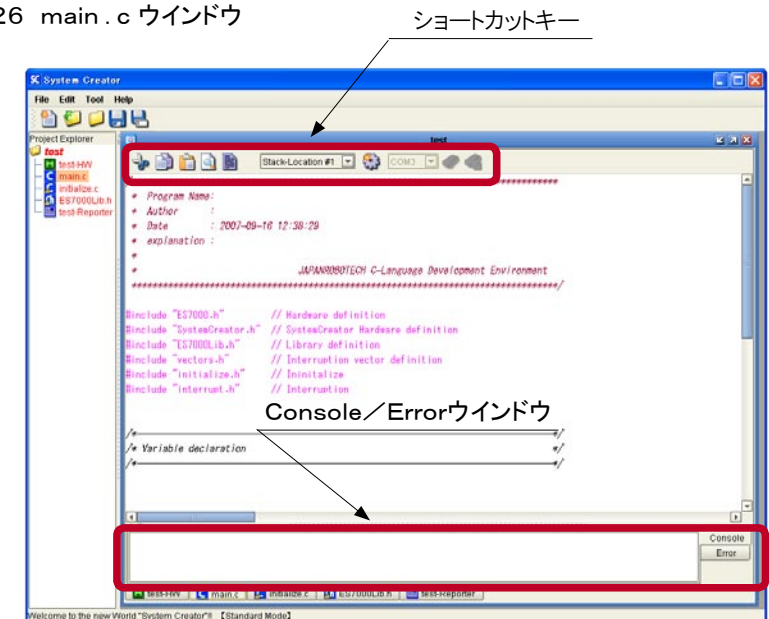
ショートカットキー （図1-26）をクリックします。

（2）Console／Errorウインドウ（図1-26）へ［Compile Complete（図1-27）］が表示されます。

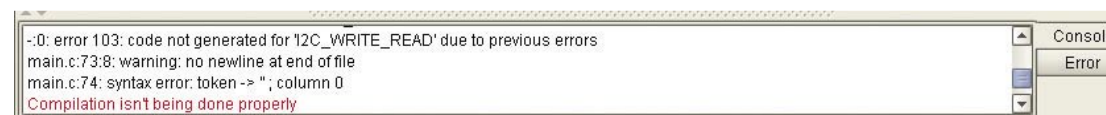
コンパイルが正常に完了したことを表します。

（3）図1-28の［Compile Error］ウイザードが表示された場合は、プログラムに「誤り」がありますので、「了解」をクリック後、プログラムの「誤り」を修正（デバック：P26参照）を行い、再度コンパイルを行って下さい。

※コンパイルは、［Compile Complete］が表示されるまで行います。

■図1-26　main.c ウインドウ

■図1-27　Console　コンパイル完了

■図1-28　Compile／Errorウイザード

■Error 時のConsole表示　※エラーの種類などの情報が表示されます。

1-5. プログラムの転送

コンパイルにより機械語へ変換されたプログラムを実行・検証のためコントローラボードへ転送します。

転送は、以下の手順で行います。

1-5-1. プログラム転送手順

（1）パソコンが認識している COMポートを確認します。図1-29
※ COMポート番号が違う場合は、ツールバー［File］の［Preference］で設定して下さい。

（2）図1-30 のように各ケーブルで接続されていることを確認します。

※ 使用できる232Cシリアルケーブルの仕様は、ストレートです。

※ USB シリアル変換ケーブル（アダプタ）をご使用の場合は、仕様をご確認の上設定して下さい。

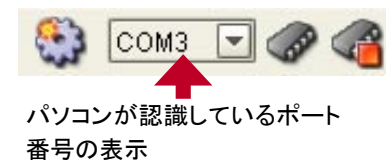
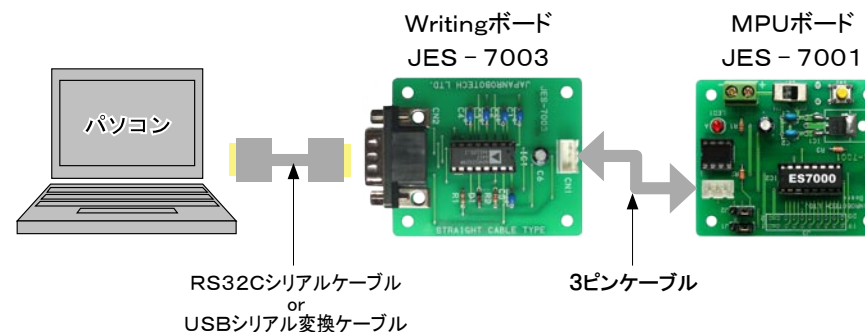
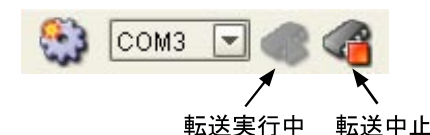
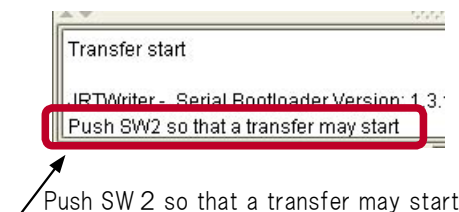
（4）MPUボードのジャンパー端子J1・J2及び各SWを以下の手順で設定します。

①電源スイッチが［OFF］であることを確認します。

②ジャンパー端子J1及びJ2を［B側］へセットします。

（5）ショートカットキー 図1-29 をクリックします。

（6）ショートカットキーが図1-31のように表示されます。
プログラム転送実行中を表します。

（7）①［Flash Writer Mode］を［ GUI ］に設定している場合は、
Consoleウインドウへ
「Push SW2 so that a transfer may start」 図1-32が表示されます。

②［Flash Writer Mode」を［Console］に設定している場合は、
［Flash Writer］が表示され、
「Push SW2 so that a transfer may start
Waiting for MPU reset ACK ...」図1-33が表示されます。

（8）電源スイッチを［ON］にします。図1-34

（9）MPUボードの［SW2］を押します。 図1-34

（10）［Console／Error］ウインドウ 最下行へ
「Memory programmed：100% Transfer complete」が表示されます。

…以上で転送完了です。

1-6. プログラムの実行：MPUボードの操作

プログラムを実行します。コントローラボードの各スイッチを以下の手順で操作します。

（1）電源スイッチが［OFF］になっていることを確認します。（［OFF］にします。）

（2）［J1］及び［J2］を［A側］にします。

（2）電源スイッチを［ON］にします。
※電源スイッチを［ON］にするとプログラムが実行されます。

1-7. コンパイルエラーの修正（デバック）

プログラムに誤りがあり、コンパイルエラーとなった場合は、以下の手順で修正を行います。

1-7-1. 修正の手順

（1）Console／Errorウインドウの［Error］タグをクリックします。図1-35
※エラー箇所の検索・修正を行うためのウインドウです。

（2）表示されたエラー箇所（各一行に表示　図1-36）をクリックすると誤りのある箇所、または誤りと関連する箇所（エディタ部）にカーソルが移動します。

（3）誤りを見つけ修正します。
※エラー箇所が複数行にわたる場合は、1行目から順番に行います。

（4）一行修正が完了したら、コンパイルします。
［Compile Complete］となれば完了です。
Compile できない場合は、エラー箇所がなくなるまで、（2）～（4）を繰り返します。

1-7-2. 編集（記述）時の注意点

●文字入力は、コメント表示以外はスペースに至るまで全て半角英数字で入力します。
※ 慣れない間は、コメント部も省略することを推奨します。

● { }、; の抜け、l（エル小文字）とI（アイ大文字）、0（ゼロ）とO（オー大文字）の間違い。

●使用した関数の綴りの誤り。
※ 誤りのないプログラム作成のためには出来るだけコピー＆ペーストを利用しましょう。
その他デバックは、根気よく行いましょう。

■図1-35　Errorタグ

■図1-36　Error関連部の表示

1-8. 保存プロジェクトの開き方

保存したプロジェクトを開く場合は、以下の手順で行って下さい。

（1）ツールバー［File］→［Project Open］をクリックします。

または、ショートカットキー をクリックします。

（2）［Open Project］ダイアログが表示されます。図1-37
※［Preference］で設定した保存先フォルダが優先表示されます。

（3）開きたいプロジェクト名の［.xml ファイル］を選択し、［開く］をクリックします。
図1-38

（4）［Project Explorer」とプロジェクトファイル切替用タグへプロジェクトファイルが表示されます。

1-9. 複数のプロジェクトの開き方

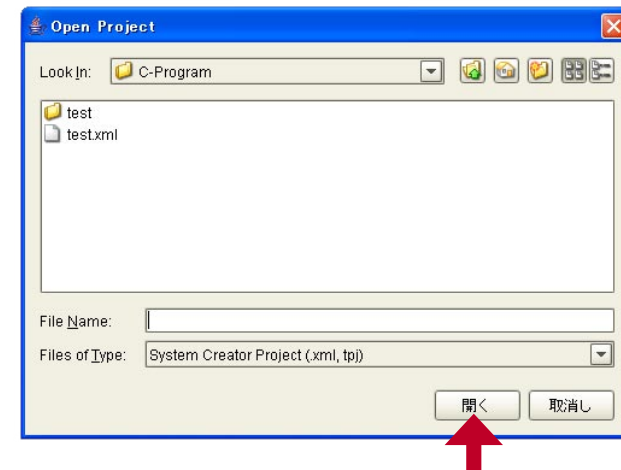
同一ウインドウ上に複数のプロジェクトを開くことが可能です。
過去に作成したプログラムを参考にしたり、プロジェクト間のコピー＆ペーストも可能です。

1-9-1. 新規プロジェクトを追加する

P14「1-2. プロジェクトの開き方 」の手順と同様の手順で開きます。

1-9-2. 保存したプロジェクトを追加する

P27「1-8. 保存プロジェクトの開き方」の手順と同様の手順で開きます。

■図1-37 ［Open Project］ダイアログ

■図1-38 .xml ファイル

●⚠注 意●

●タイルプログラムもご使用の方はタイルプログラムの .xml プログラムと混同しないよう別々のフォルダへ保存することをお勧めします。

1-10. プロジェクトファイルのウインドウサイズ変更

1-10-1. ウインドウサイズの最大化・最小化

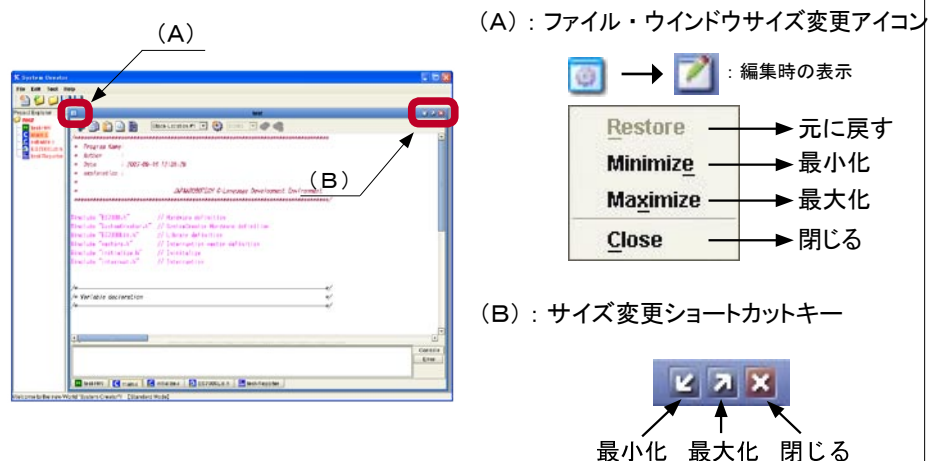
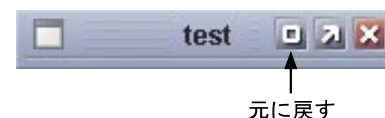
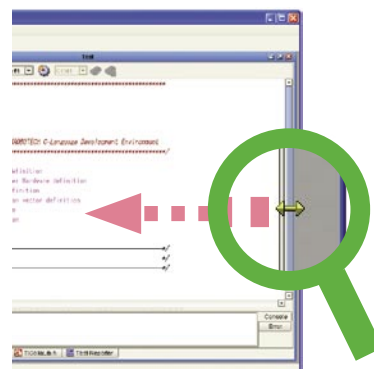
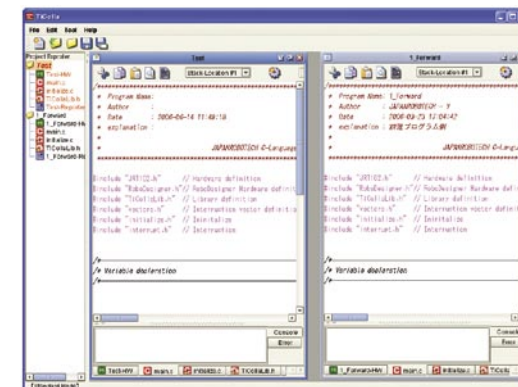
複数のプロジェクトを使用して編集する場合、プロジェクトの表示切替を行います。

（A）サイズ変更アイコン及び、（B）ショート カットキーで任意に変更が可能です。
図1-39・40

1-10-2. ウインドウサイズの任意変更

パソコンの一般的なウインドウ操作や各種アプリケーションのウインドウ操作と同様にファイル・ウインドウのサイズを任意の大きさに変更、複数のファイル・ウインドウを同時に表示させることも可能です。

マウスポインタを図1-38の位置ファイル・ウインドウ画面端（上下左右）へ移動させるとマウスポインタが矢印 ⇔ へ変わります。矢印のままクリックし、左右にドラッグするとサイズの変更が可能です。図1-41・42

■図1-39　プロジェクト別ファイル・ウインドウ切替操作

■図1-40　ファイル・ウインドウ最小化時の表示

■図1-41　サイズの任意変更

■図1-42　プロジェクト別ファイル・ウインドウの並列表示

1-11.［Project Explorer」の操作

「Project Explorer」で操作可能な内容は以下の通りです。

1-11-1. プロジェクト及びプロジェクトファイルの切替機能

複数のプロジェクトを開いている場合は、表示プロジェクトの切替が可能です。
プロジェクトファイル切替用タグ同様にクリックするとファイルの切替が可能です。

1-11-2. 開かれているプロジェクトの表示

プロジェクトの内容表示／内容非表示の切替　図1-43
フォルダをダブルクリックするとフォルダの開閉が可能です。

1-11-3. ファイルの保存・追加・クローズ機能

以下の操作は、フォルダ部の「右クリック」から表示されるポップアップメニューより選択設定して行います。図1-44

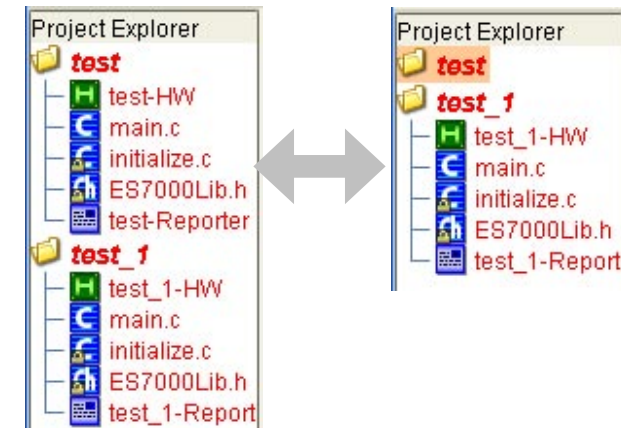
1）add File

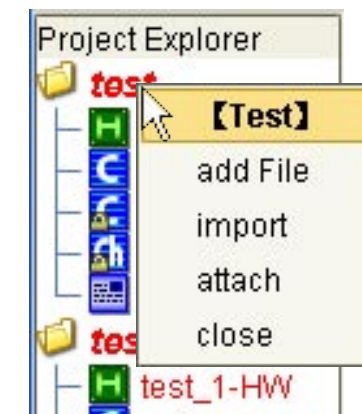
サブルーチン等エディタ及びヘッダーファイルの追加作成機能です。

【手　順】

（1）［add File］をクリック
（2）［New Project / File Wizard］の表示
（3）［Resource Name」の記入
（4）［Finish］をクリック
（5）プロジェクト内に新規Cエディタファイル及びヘッダーファイルが作成されます。

■図1-43

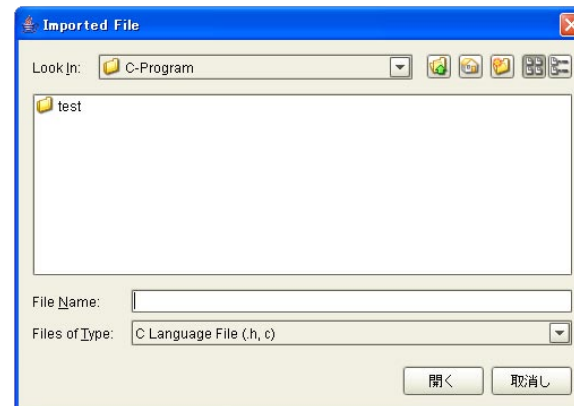
フォルダ部をダブルクリックすることで表示の変更が可能です。フォルダに収納されます。

■図1-44

フォルダ部を「右クリック」することでメニュー表示が可能です。

2）import

他のプロジェクトで作成されたサブルーチンを現在開かれているプロジェクトへ読み込む機能です。読み込みはCエディタファイルとヘッダーファイルの2つをそれぞれ読み込まなければなりません。また読み込まれたファイルは内容の編集が可能です。

【手　順】

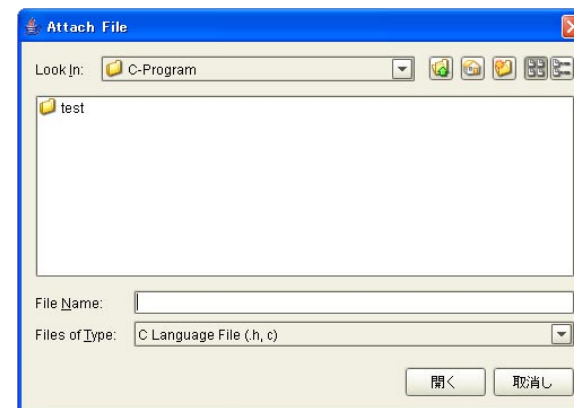
（1）［import］をクリック
（2）［Imported File］ダイアログ（図1- 45）の表示
（3）任意のフォルダまたはファイルを指定
（4）［開く］をクリック
（5）プロジェクト内に任意のファイルが追加されます。

■図1- 45 ［Imported File］ダイアログ

3）attach

［import］と同様に他のプロジェクトで作成されたサブルーチンを現在開かれているプロジェクトへ読み込む機能です。
読み込みはCエディタファイルとヘッダーファイルの2つをそれぞれ読み込まなければなりません。
［import］と異なり、読み込まれたファイル内容の編集（変更）ができません。

【手　順】

（1）［attach］をクリック
（2）［Attach File］ダイアログ（図1- 46）の表示
（3）他のプロジェクトの開きたいファイルを指定し、［開く］をクリックします。

■図1- 46 ［Attach File］ダイアログ

4）close

プロジェクトを「閉じる」機能です。

【手　順】

（1）［close］をクリック
（2）［Project Save Comfirm］ウイザード（図1-47）の表示
（3）「保存［はい］」または、「保存しない［いいえ］」を任意に選択し、プロジェクトを閉じます。

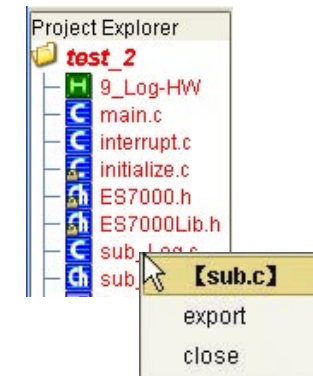
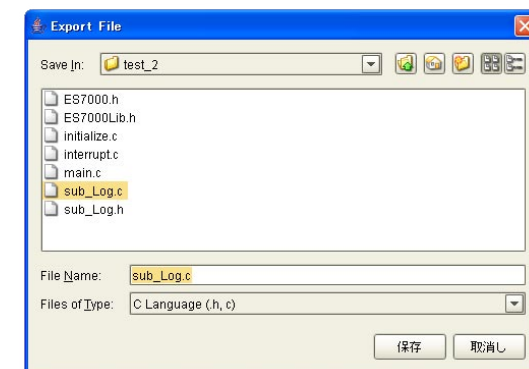
■図1-47 ［Project Save Comfirm］ウイザード

5）export

プロジェクトで作成された各ファイルを個別に保存する機能です。

【手　順】

（1）保存したいファイルを「右クリック」
（2）ポップアップメニューの表示(図1-48)
（3）［export］をクリック
（4）［Export File］ダイアログ（図1-49）の表示
（5）任意の保存先、任意名へ設定
（6）［保存］をクリックします。

■図1-48 ［export］

■図1-49 ［Export File］

1-12. 検索・置換

プログラムの大きさにもよりますが、綴りや記号の誤り等を発見し、その関数（単語）や記号を一つ一つ手直しするのは大変な作業です。
誤りの原因が明白である場合は、検索・置換機能を利用することで、作業の効率化が図れます。

1-12-1. 検索・置換の手順

（1）ファイルウインドウの

ショートカットキー　図1-50 をクリックします。

（2）［Text Search］ダイアログが表示されます。図1-51

（3）（A）には、検索したい関数（単語）や記号（スペースの半全角も含みます）を入力します。図1-51では、「STOP」と入力しています。

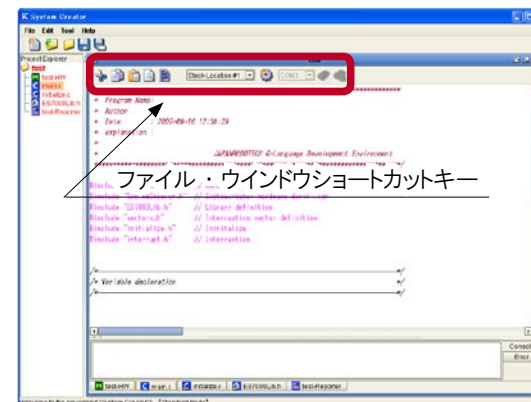
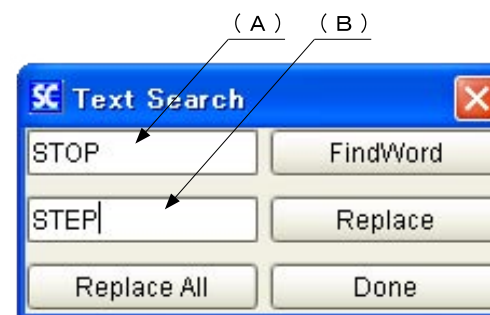
（4）以下の機能で検索・置換します。

【FindWordキー】：（A）に記入された文字がある箇所をセレクト（選択）した状態で表示します。図1-52

【Replaceキー】：（A）に記入された文字の箇所を（B）の文字へ置き換えます。

【Replace All l キー】：（A）に記入されたテキスト内の文字の箇所を（B）の文字へ一括で置き換えます。

【Doneキー】：終了、完了キーです。ダイアログを閉じます。

■図1-50

■図1-51 ［Text Search］ダイアログ

■図1-52 ［FindWord］実行時の結果表示例

```
init();          //Initialization

        MOTOR_SPD( 1, MOTOR_STOP_MID2);

while(1){
```

検索した文字がセレクトされた状態

1-13. データロガー

センサから取得したアナログ電圧の変化や判りにくいマイコン内部の変数値を時系列にリスト化、ラインチャート表示が可能です。

この機能は、ソフトウェア上で機能します。このため、この機能を使用するためには、必ず専用のプログラムが必要となります。
プログラムについては、プログラミングソフト SC7000 内の［Sample Project D_Log］または、P62 の［サンプルプログラム D_Log］を参照してください。

1-13-1. ファイル・ウインドウの開き方

（1）ツールバー［File］→［New Project］

または、ショートカットキー をクリックします。

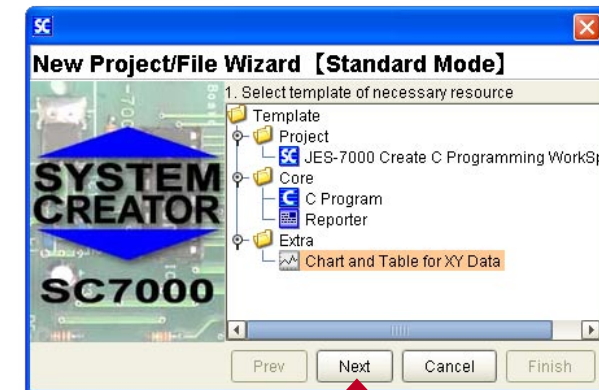
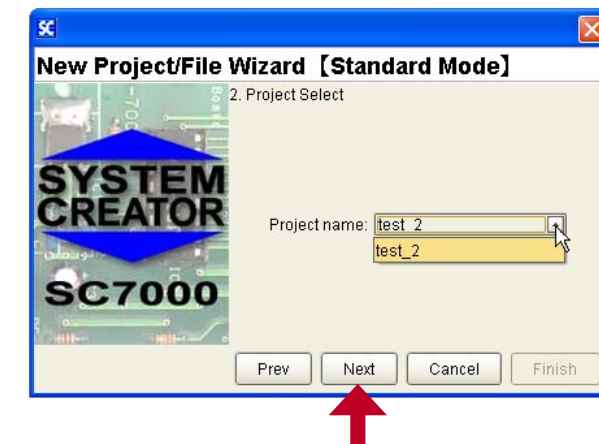
（2）［New Project / File Wizard］が表示されます。図1-53

（3）フォルダ［Extra］のファイル

Chart and Table for XY Data

を選択し、［Next］をクリックします。

（4）［2. Project Select］の表示。セレクタで「データロガーを取り込みたいプロジェクト」を任意に選択し、［Next］をクリックします。図1-54

■図1-53 ［New Project / File Wizard］

■図1-54 New Project / File Wizard ［2. Project Select］

（5）［3. Enter resource name］の表示。
任意のファイル名（使用しているプロジェクト名以外の名前）を入力し、[Finish]
をクリックすると、プロジェクトに追加されます。　　　　　　　　図1-55

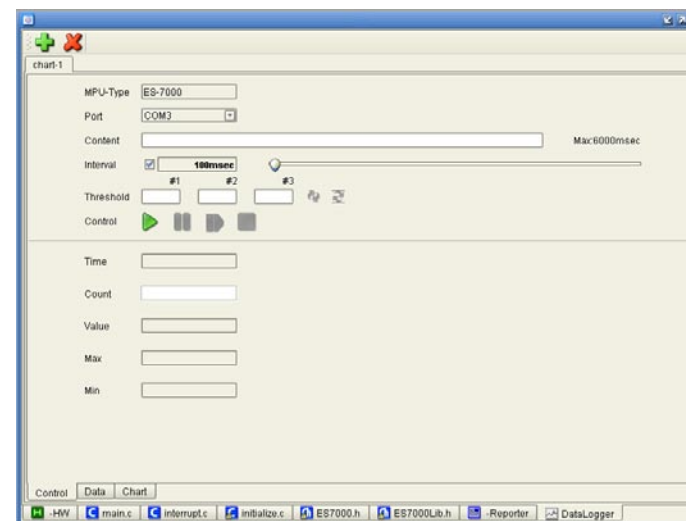
（6）［Project Explorer］または、「ファイル・ウインドウ切替タグ」の

～（～はファイル名）をクリックすると

図1-56　のデータロガーファイル・ウインドウが表示されます。

…以上で終了です。

■図1-55　New Project / File Wizard ［3. Enter resource name］

■図1-56　DataLogger ファイル・ウインドウ

1-13-2. データロガー操作各部解説　図1-57

1）Control ウインドウ解説

（1）ウインドウ切替タグ　図1-58

［Control］：ロガー操作部

［Data］　：データ取得日時別データリスト表示

［Chart］　：ラインチャート（折れ線グラフ）表示

の切替を行います。

（2）データ別ウインドウ切替タグ及び、
（3）リスト及びチャートの追加・削除ショートカットキー　図1-59

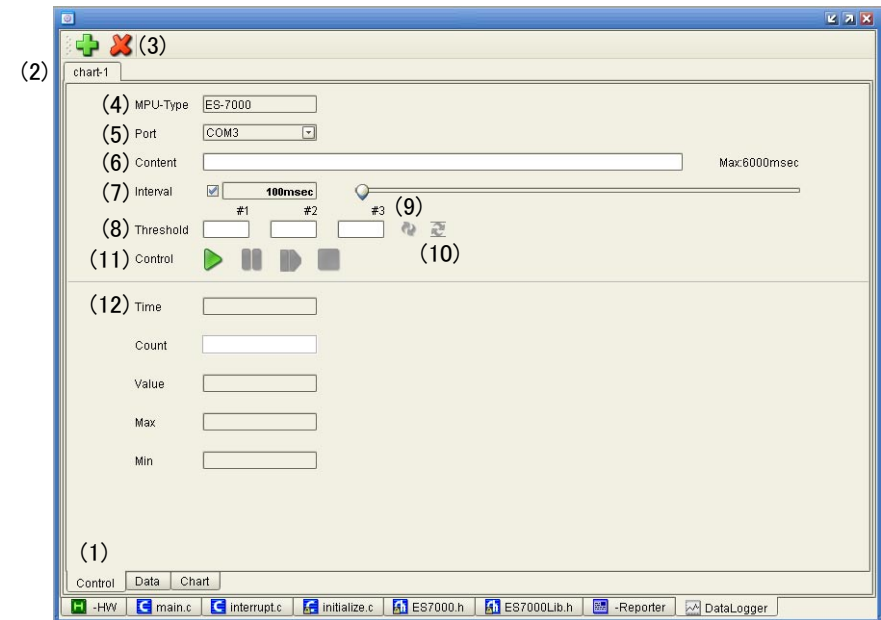
同一プログラム、同一ウインドウ上で、複数のデータの記録が可能です。

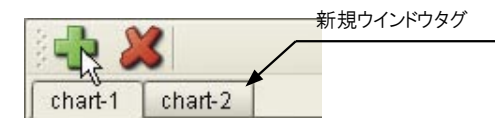
①(3)ショートカットキー をクリックすると、新規リスト及びチャートを追加します。

②(3)ショートカットキー をクリックすると、開かれているリスト及びチャートを削除します。

（4）MPU Type

使用マイコンの型式が表示されます。

■図1-57　Data Logger ファイル・ウインドウ

■図1-58　ウインドウ切替タグ

■図1-59　データ別ウインドウ切替タグ及び追加・削除ショートカットキー

（5）Port

232C シリアル通信ポートの表示。
「Preference」で設定されたポートが表示されます。

（6）Content

表題記入用テキストフレームです。

（7）Interval

データ取得時間間隔の設定を行います。

【設定範囲】

10msec（0.01秒）～60000msec（1 分）, 10msec 毎設定可。

【設定方法】 図1-60

①Interval を設定する場合は、図1-60 □（チェック欄）へチェックを入れます。

②スライドスイッチのツマミを右へドラッグするとアップされます。

③スライドスイッチ上でクリックするとアップされます。
（1クリック毎 ： 10msec Up）

●⚠注 意●

※［Interval］の時間設定は、小さくなれば、小さくなるほど一定時間に使用するメモリ使用量が大きくなり、オーバーフローを起こす場合があります。オーバーフローを起こした状態では、正しいデータ取得の保障はできません。適正な時間設定でご使用下さい。

■図1-60 Interval 設定スイッチ

(8) Threshold

ラインチャートへ目安となるライン（３種）を挿入することができます。

【操作方法】

①＃1・＃2・＃3へChartで表示された最大値～最小値の間で数値を入力します。
図1－61

②（9）Chart Refleshキー または、（10） をクリックすると、

Control部で 設定した数値でChart部へラインが表示されます。 図1－62

③ラインの位置を変更する場合は、入力した数値を任意に変更し、(9)または(10)のChart Refleshキーをクリックします。

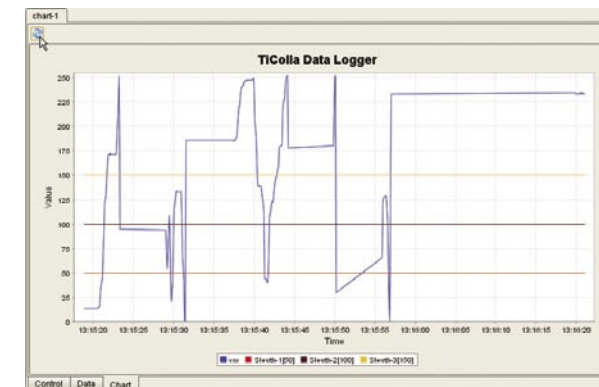
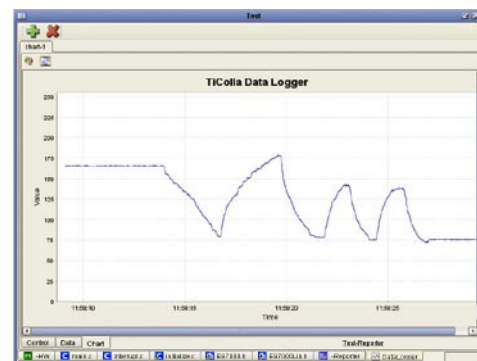
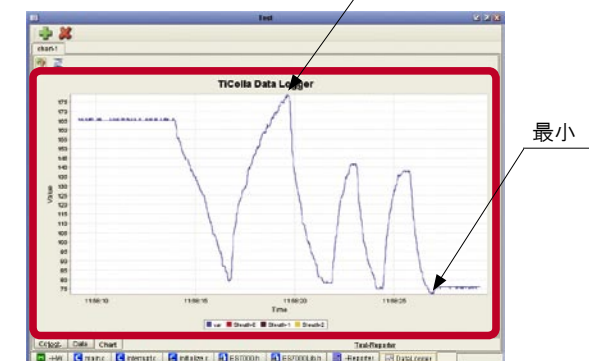
（9） Chart Refleshキー

①［Threshold］で設定した数値のラインを表示させせます。
※（8）Thresholdを参照下さい。

②Chart 表示範囲を最大値から最小値の範囲でフィットさせて表示します。
図1－63・64

③赤枠画面の最大化・最小化の切り替えを行います。図1－64
※Chart画面にも同じキーがあります。

■図1－61 ［Threshold］

■図1－62 Chart ウインドウ

■図1－63 データ取得後の初期Chart 画面

■図1－64 （9）Chart Reflesh クリック時

（10） Chart Reflesh キー

①［Threshold］で設定した数値のラインを表示させます。
※（8）Threshold を参照下さい。

②Chart 表示範囲 0～255 の範囲で表示します。

③赤枠画面の最大化・最小化の切り替えを行います。図1－64
※Chart 画面にも同じキーがあります。

（11）Control

①Logging Startキー ：データ記録を開始します。

②Logging Pauseキー ：データ記録を一旦停止します。

③Logging Restartキー ：データ記録を再開します。

④Logging Stopキー ：データ記録を停止します。

（12）各種データ表示

① Time ：1 データの記録年月日・時間表示

② Count ：記録中のデータの数を表示

③ Value ：最新データの値を表示

④ Max ：記録データの中で最大値を表示
※［Project Open］で保存プロジェクトを開いた場合は表示されません。

⑤ Min ：記録データの中で最小値を表示
※［Project Open］で保存プロジェクトを開いた場合は表示されません。

2) Data ウインドウ解説　図1-65

(1) Time（列）

データを記録した年月日時間を表示します。

(2) Ver（列）

記録された値を表示します。

(3) Data Save キー

Data Saveキーをクリックすると、取得データが .csv (Excel) ファイルで保存されます。

●⚠注 意●

取得したロガーデータは、そのプロジェクトのみを保存しても、ロガーデータの保存はできません。取得したロガーデータを保存する場合は、必ずData Save キーによる保存を行って下さい。

■図1-65　Data ウインドウ

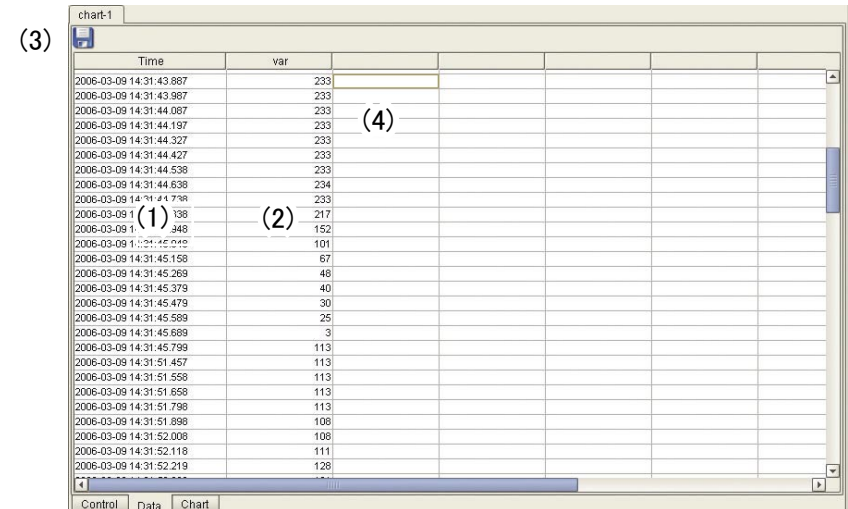
【保存の仕方】

①Data Saveキーをクリックすると[Save Logger Data]ダイアログが表示されます。

② [Preference] で設定した保存先フォルダを指定するか、任意の「保存先（マイドキュメントや C:、D:）を指定し、任意名の .csv ファイルを保存します。

(4) 記録したデータ計算機能　例) オフセットの調整等に使用します。

【操作方法】

①空欄のセルをクリックすると、「入力」ダイアログが表示されます。図1-66
　※「var」は値を表します。
②「計算させたい数式」を入力し、「了解」をクリックします。
　入力例)「var－50」、「var＋50」、「var＊50」、「var／50」、「0-var」等
※「var」を消去して入力すると、入力した文字が表示されます。

■図1-66　入力ダイアログ

3) Chart ウインドウ解説

（1）ラインチャート画像の保存

【操作方法】

①画面上で「右クリック」すると、メニュー(図1-67)が表示されます。

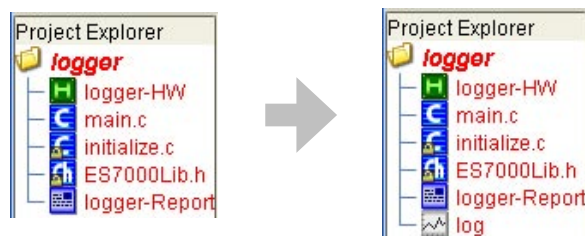
②［Save as …］をクリックすると、［保存］ダイアログの表示されます。
保存先を指定して任意名入力後、［保存］をクリックします。
※.pngイメージデータで保存されます。

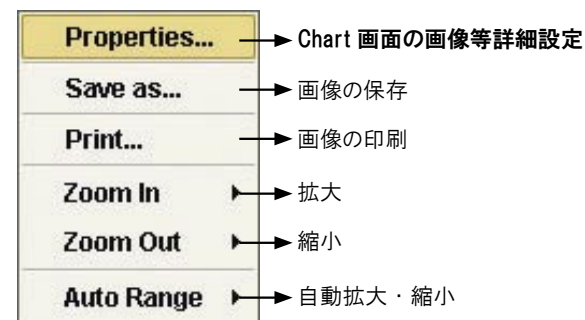
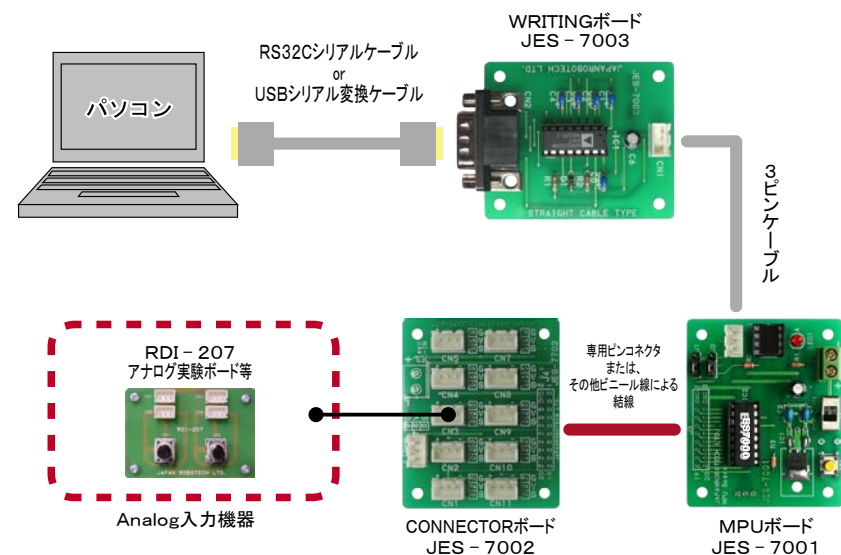
(2) Chart Refleshキー

※Control (9)､(10) を参照して下さい。

1-13-3. データロガー使用手順

（1）データロガー用のプロジェクト（HW及びmain.c 作成済み）を作成します。
※サンプルプログラム「D_Log」を参照ください。

（2）プロジェクトにデータロガーファイル(Chart and Table for XY Data)を追加します。
※P33「1-13.データロガー 1-13-1.ファイル･ウインドウの開き方」参照

（3）図1-68 のように接続し、準備します。
※Control ウインドウ［Port］で一応ポートの確認を行って下さい。
※P36「（7）Interval」を参照し、適正な時間設定を行って下さい。

（4）MPUボードの［J1］及び［J2］を［B側］にし､ロガー用のプログラムをP24「1-5-1.プログラム転送手順」の手順に従い転送します。

（5）MPUボードの［J1］及び［J2］を［A側］にします。

（6）MPUボードの電源スイッチを［ON］にし､プログラム実行後､［J2］を［B側］にします。
※サンプルプログラム「D＿Log」では、誤動作防止のためアナログデータ取得前に［J2］切り替え用のディレイ・タイム（CN7～CN10へ接続されたLEDの20回点滅）を記述しています。

（7）プログラム転送が完了したらロガー・ファイル（Chart and Table for XY Data）・ウインドウ（Control ウインドウ）を開きます。

（8）ロガーControl ウインドウの［Logging Start キー］をクリックします。
データ取得を開始します。　※その他の設定は､P35「1-13-3.データロガー操作各部解説」を参照下さい。

（9）[Count] を見ながら必要なデータ数を取得したら、[Logging Stop キー］をクリックします。
…以上でデータ取得操作完了です。

（10）取得したデータを保存する場合は、P39［(3) Data Save キー］をクリックし、任意名で保存します。

（11）取得データ保存後、データロガーのプロジェクト自体を保存します。
※保存したプロジェクトを開くと、データロガーファイルのロガーデータも表示されます。この場合、データロガーファイルの［Control］部の［Max］及び［Min］の数値表示はありません。

（12）同じデータロガープロジェクトで別のロガーデータを取る場合は、［Project Explorer］上でデータロガーファイルを［右クリック］から［close］し、新規に［データロガーファイル］を作成して下さい。
※保存する場合は .csv ファイル、プロジェクト共に「別名で保存」します。

※その他の設定は､P 35「1-13-3.データロガー操作各部解説｣参照下さい。

…以上で操作完了です。

●⚠注 意●

●データ送受用のシリアルケーブルは、接続したまま使用します。
●図1-68では、アナログ実験ボード等（赤破線枠）を接続していますが、取得するデータの内容によってはその限りではありません。任意に準備して下さい。
●[J1]と[J2]の詳細については､P43［2-1-1. コントローラボード各部名称・解説］⑤ J1ジャンパー 及び ⑥J2ジャンパーを参照してください。

重　要

■データロガー使用手順解説

①この機能は、ソフト上で行う機能です。データロガーを使用する場合は、データロガー用のプログラムが必要です。プログラムについては、P62「3-5. データロガー（通信）サンプルプログラム　D＿Log」を参照してください。

②手順（5）及び（6）について

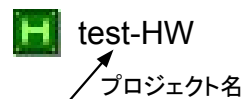
●（5）の［J2］を［A側］にすることにより､データロガー用プログラムの［実行待ち］状態にします。（6）の電源SWをONにするとデータロガー用プログラムが実行されデータ取得を行います。パソコンへこの取得データを転送（通信）するためには、必ず［J2］を［B側］にする必要があります。

●サンプルプログラム「D＿Log」中、アナログデータ取得のためのサブルーチン呼び出し前に「LED20回点滅」を行わせています。これは､取得データを送信しながら、［J2］の切替を行った場合、そのタイミングによっては、「データ送信途中で切り替わる可能性」があり、誤動作につながる恐れがありますので、切り替え用の待ち時間（LED20回点滅中に［J2］を切り替えるための時間）を記述しています。

●この切り替え用「待ち時間」のプログラムは、サンプルプログラム D＿Log の記述例でも良いですし､「i<20」もしくは、DELAY_SEC の値を変更することにより送信までの時間を変更して使用することも可能です。

2. プロジェクト解説

プロジェクトを構成する環境は、以下の内容です。
　　は［Advanced］モード時表示、その他は［Standard］、［Advanced］各共通表示です。

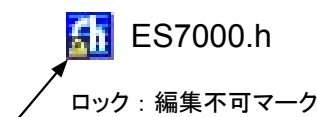
① HW（ハードウェア設計、回路設計）
●MPUボードのポート（CN1～CN11）の入出力を任意の機器で設計します。
※詳細は、P43「2-1. HW解説」参照して下さい。

② main.c（主エディタ部）
●制御プログラムを記述する主エディタ部です。グローバル変数・自作関数宣言、自作関数の定義、制御プログラムの記述を行います。

③ Interrupte.c（割り込み処理エディタ部）

●割り込み処理を記述するエディタ部です。

④ Initialize.c（ハードウェア設定反映ファイル）
●ハードウェア設定による内容が反映されたファイルです。
※このファイルは編集できません。

⑤ ES7000.h

● HW 画面にある入出力コネクタ（ポート）の定義を行っています。
※このファイルは編集できません。

⑥ ES7000Lib.h（関数ライブラリ）
●専用関数ライブラリーです。エディタ部で使用する制御用の各種専用関数が定義されたヘッダーファイルです。
※このファイルは編集できません。

⑦ Reporter（ワープロ、リポーター機能）
●簡易ワープロ機能です。研究・学習時の記録やレポート、取得データのメモ等自由に使用できます。

2-1. HW（ハードウェア設計）解説

HW（ハードウェア設計）・ウインドウで設計された回路は、ソースプログラムである［Initialize . c］ファイルへ自動的に反映されます。

2-1-1. コントローラボード各部名称・解説

（1）DC - IN　電源端子

■電源電圧／DC6V ～ DC12V

（2）SW1：RUN　電源及び実行スイッチ

■回路電源 ON ／ OFF スイッチ 兼 プログラム実行スイッチ
※電源 ON と同時にプログラムが実行されます。

（3）SW2：IRQ

■プログラム転送時の［トリガースイッチ］です。
※割り込みプログラム実行時の［IRQ キー・スイッチ］としても使用可能です。
■SW2付属の2つの端子（図2－1においてSW2の下）は、SW2のパラレル端子になっていますので、ビニール線等で延長し、外部へSWを設けることも可能です。

（4）データ通信用入出力端子

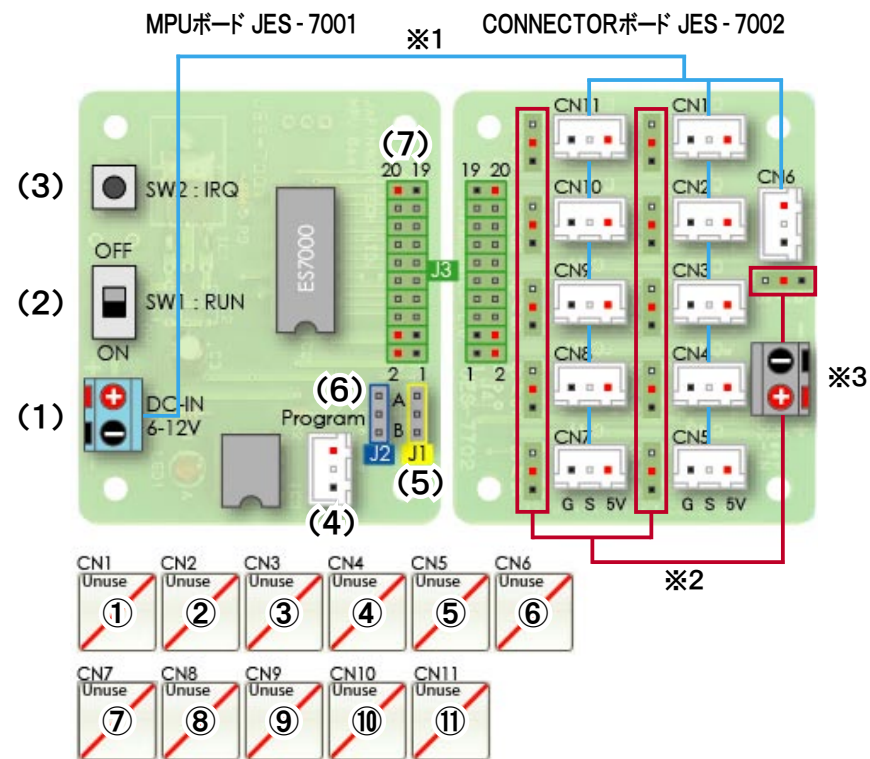
■WRITINGボード JES－7003 とのデータ通信用の端子です。

（5）J1　ジャンパー1

■プログラム転送及びプログラム実行時に MPUとJ3（入出力端子）の信号回路のON（接続）とOFF（切断）を切り替えます。
※プログラム実行時　→［A側（ON)］
　プログラム転送時　→［B側（OFF)］　ヘジャンパーソケットを挿し替えます。

（6）J2　ジャンパー2

■プログラム転送及びプログラム実行の各モードを切り替えます。
※プログラム実行時　→［A側（AutoRun モード)］
　プログラム転送時　→［B側（ダウンロード：通信モード)］
　ヘジャンパーソケットを挿し替えます。

■図2-１ HW（ハードウェア）設計図　初期設計画面

※1 ━━ ：JES－7002 の入出力用コネクタの電源ライン。JES－7001 の回路（レギュレートされた）電源（DC5V）です。

※2 ━━ ：JES－7002 の各入出力用コネクタへ近接する各入出力用端子（ランド：▪▪▪）の電源ライン。制御する入出力機器の電源電圧に合わせ、任意に供給します。（電源端子※3を使用します。）
ランド　白：S（シグナル）、赤：V（＋）、黒：G（グランド）

（7）J3　J3端子

■外部入出力機器制御用 信号入出力ポート及び電源用端子
※各端子の仕様は、HW設計画面の右下記載されています。図2－２

■信号入出力ポート　CN1 ～ CN11
信号入出力ポートは、全11ポートです。
この各ポートを Digital Input／output，Analog Input 等使用したい外部入出力機器に合わせ設計を行います。
設計は、図2－３ 赤枠内（タイルパレット）のタイルを 図2－１ の ① ～ ⑪ のマスへ使用したい機器や信号の入出力を表すタイルをドラッグ＆ドロップで行います。

■CONNECTORボード JES－7002 は、拡張用ボード（別売）です。
［J3端子］をポート別のコネクタ（G・＋5Vライン付）へ変換しますので、仕様に合った入出力機器の接続に大変便利です。

① CN1 ・ ② CN2・ ⑤ CN5及び ⑦ CN7～⑩ CN10

■各ポート［Digital Input／Output］の選択・設計可能です。

③ CN3及び ④ CN4

■各ポート［Digital Input／Output］及び、［Analog Input］の選択・設計が可能です。

⑥ CN6

■［Digital Input／Output］、［Analog Input］及び、［PWM信号］、［COM（通信：DataLogger用）］の選択・設計が可能です。

●⚠注 意●

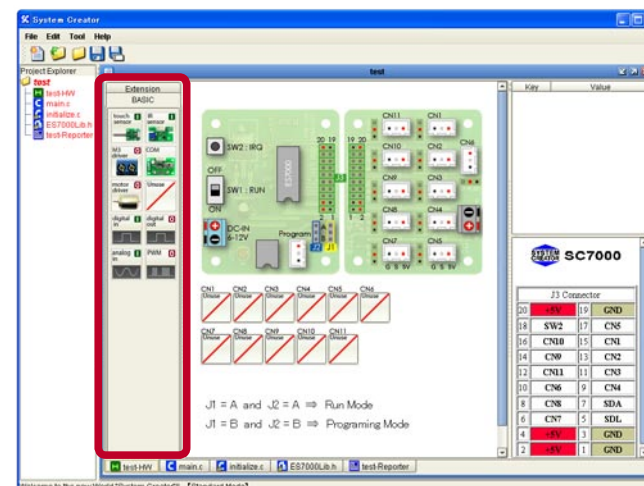
CN6をセンサ等の入力機器で使用する場合は、プログラム実行時以下の手順で行ってください。プログラムが正しく実行されない場合があります。
①電源SWをOFF→②J1を［Ｂ側］へする→③電源SWをON（プログラムの実行）→④J1を［Ａ側］へする

⑪ CN11

■［Digital Input／Output］、［Analog Input］及び、［PWM信号］の選択・設計が可能です。

■図2－２　J3　端子仕様

J3 Connector			
20	+5V	19	GND
18	SW2	17	CN5
16	CN10	15	CN1
14	CN9	13	CN2
12	CN11	11	CN3
10	CN6	9	CN4
8	CN8	7	SDA
6	CN7	5	SDL
4	+5V	3	GND
2	+5V	1	GND

■図2－３　HW（ハードウェア）設計・ウインドウ

2-1-2. タイル解説

タイル名・他	仕様及び注意	タイル名・他	仕様及び注意
① touch sensor （タッチセンサタイル） touch sensor 【使用可能ポート】 CN1～CN11	タッチセンサ RDI－201を使用する際配置します。 ※［digital in］と同じです。	⑥ Unuse 【使用可能ポート】 Unuse CN1～CN11	任意のポートを使用しない場合に配置します。 その他の配置されたタイルを削除する場合も使用します。
② IR sensor （アナログ赤外線センサタイル） IR sensor 【使用可能ポート】 CN3・CN4 CN6・CN11	アナログ赤外線センサ RDI－202を使用する際配置します。 ※［analog in］と同じです。	⑦ digital in （ディジタルインタイル） digital in 【使用可能ポート】 CN1～CN11	任意のポートをディジタルインで使用する場合配置します。 ※タッチセンサタイルと同じです。
③ M3 driver（M3 端子タイル） M3 driver	※［digital out］と同じです。	⑧ digital out （ディジタルアウトタイル） digital out 【使用可能ポート】 CN1～CN11	任意のポートをディジタルアウトで使用する場合配置します。
④ COM（データ通信タイル） 【使用可能ポート】 COM CN6	データロガー機能などパソコンと通信によりデータを送受する際配置します。	⑨ analog in （アナログインタイル） analog in 【使用可能ポート】 CN3・CN4 CN6・CN11	任意のポートをアナログインで使用する場合配置します。 ※アナログ赤外線センサタイルと同じです。
⑤ motor driver motor driver	※［digital out］と同じです。	⑩ PWM（PWM信号タイル） PWM 【使用可能ポート】 CN6・CN11	PWM信号出力に使用する場合配置します。 ※モータ速度制御等へ利用することができます。

2- 2. main.c 解説

```c
/*****************************************************************************
 *  Program Name:
 *  Author      :
 *  Date        : 2006-03-11 07:47:58
 *  explanation :
 *
 *
 *                         JAPANROBOTECH C-Language Development Environment
 *****************************************************************************/

#include "ES7000.h"          // Hardware definition
#include "SystemCreator.h"   // SystemCreator Hardware definition
#include "ES7000Lib.h"       // Library definition
#include "vectors.h"         // Interruption vector definition
#include "initialize.h"      // Ininitalize
#include "interrupt.h"       // Interruption


/*---------------------------------------------------------------------------*/
/* Variable declaration                                                      */
/*---------------------------------------------------------------------------*/


/*---------------------------------------------------------------------------*/
/* Prototype declaration                                                     */
/*---------------------------------------------------------------------------*/


/*---------------------------------------------------------------------------*/
/*  main                                                                     */
/*---------------------------------------------------------------------------*/

void main(void) {

        init();         //Initialization


        while(1){

                COPCTL = 0;
        }
}
```

プログラム名
制作者名
日付
簡易説明

インクルード文：プログラムに必要な各種ヘッダーファイルを呼び出します。
※この部分は、プロジェクトを立ち上げた際、自動的に作成されます。

●⚠注 意●
インクルード文の変更・削除は行わないで下さい。

グローバル変数宣言：プログラムで使用する変数を記述します。

関数プロトタイプ宣言：自作関数を記述します。

main 関数定義：制御プログラムを記述します。自作関数定義等も行います。

処理が1回のみの場合は、「init();」と「while(1)」の間に記述します。

ループで何回も繰り返すような処理は「while(1)」のループ内に記述します。
ループを記述する場合は、「COPCTL=0;」をループ中に記述してください。

※ COPCTL：コンピュータ正常動作モジュール・レジスター
このレジスターの値が一定期間クリアされないとマイクロプロセッサーは暴走したと判断し自動的にリセットがかかります。

2-3. Interrupt.c 解説

```
/*************************************************************************
 *  Program Name: Interrupt process Program.
 *  Author      :
 *  Date        : 2006-03-11 07:47:58
 *  explanation :
 *
 *
 *                         JAPANROBOTECH C-Language Development Environment
 *************************************************************************/

#include "ES7000.h"          // Hardware definition
#include "SystemCreator.h"   // SystemCreator Hardware definition
#include "ES7000Lib.h"       // Library definition

/*----------------------------------------------------------------------*/
/* Variable declaration                                                 */
/*----------------------------------------------------------------------*/


/*----------------------------------------------------------------------*/
/* Prototype declaration                                                */
/*----------------------------------------------------------------------*/


/*----------------------------------------------------------------------*/
/* ADC_Interrupt - ADC Conversion Complete Interrupt                    */
/*----------------------------------------------------------------------*/

//#pragma TRAP_PROC
void _ADC_Interrupt(void) {
        __DI();

        __EI();
}

/*-------------------------- 以下略 ---------------------------------*/
```

インクルード文：プログラムに必要な各種ヘッダーファイルを呼び出します。

●⚠注 意●

インクルード文の変更・削除は行わないで下さい。

グローバル変数宣言：プログラムで使用する変数を記述します。

関数プロトタイプ宣言：自作関数を記述します。

割り込み処理定義

各割込み用の処理ルーチンへの記述は、

「_DI( )（割込み禁止マクロ）」

「_EI( )（割込み許可マクロ）」　の間に記述してください。

その他割り込み処理用関数

- _TOF_Interrupt(void);
- _TCH0_Interrupt(void);
- _TCH1_Interrupt(void);
- _IRQ_Interrupt(void);

2-5. Initialize.c 解説

●⚠注 意●

「Initialize.c」の編集はできません。設定の変更は「HW 画面」で行って下さい。

```
/* ***********************************************************************
 *  Program Name: Hardware is initialized.
 *
 *      (C) Copyright Japan-RoboTech Co.,Ltd All rights reserved
 * ***********************************************************************/
#include "ES7000.h"          // Hardware definition
#include "SystemCreator.h"   // SystemCreator Hardware definition
#include "ES7000Lib.h"       // Library definition

/*-----------------------------------------------------------------------*/
/*  Initialization                                                       */
/*-----------------------------------------------------------------------*/

void init(){
                PORT_INIT();
                I2C_INIT();

                //AutoCodingStart
                CN11 = 0;
                CN11DDR = 1;
                CN11PUE = 1;
                PWM2_INIT();
                CN7 = 0;
                CN7DDR = 1;
                CN7PUE = 1;
                CN8 = 0;
                CN8DDR = 1;
                CN8PUE = 1;
                CN9 = 0;
                CN9DDR = 1;
                CN9PUE = 1;
                CN10 = 0;
                CN10DDR = 1;
                CN10PUE = 1;
                TSC_TSTOP = 0;
                //AutoCodingEnd}
```

インクルード文：プログラムに必要な各種ヘッダーファイルを呼び出します。

初期設計（ハードウェア設計部）：ハードウェアコンフィギュレーターで作成された部分です。

2-5. ES7000Lib.h 解説

```
/* **********************************************************************
 *  Program Name: SystemCreator C library
 *
 *                    (C) Copyright Japan-RoboTech Ltd All rights reserved
 * **********************************************************************/

#include "ES7000.h"


/*----------------------------------------------------------------------*/
/*  Common definition                                                   */
/*----------------------------------------------------------------------*/

#define  FALSE  0
#define  TRUE   1

#define NORMAL_END      1                       // normal end
#define ERROR_END       0                       // error end
#define BIT_ON          1                       // bit on
#define BIT_OFF         0                       // bit off

#define DEGITAL_IN      0                       // degital in
#define DEGITAL_OUT     1                       // degital out

/*----------------------------------------------------------------------*/
/*  Definition related to LED                                           */
/*----------------------------------------------------------------------*/

/* LED flash */
extern void LED_FLASH(byte _flashcnt);
```

1)【LED_AFLASH 関数】：LED1 ～ 4 つを点滅するための関数です。

※引数に点滅回数を 1 から 255 の範囲の値を設定することで、任意の回数分点滅します。

例） LED_AFLASH(4);

```
/* LED DATA OUT (LSB 4bit) */
extern void LED_OUT(byte _value);
```

2)【LED_OUT 関数】：「byte_value」の下位 4bit の値を LED1 ～ 4 に表示します。

※ LSB が LED1 となり基板上の左端となるため注意が必要です。

モータ制御用の関数は、小型のロボット競技会や小型DCモータ制御などを簡単に出来るようライブラリを準備しています、一般的な正反転モータドライバ回路への接続は下記を参考にして下さい。

●モータ1正転信号：CN7　●モータ1反転信号　：CN8　●モータ2正転信号　：CN9　●モータ2反転信号　：CN10　●モータ速度制御信号　：CN11

```
/*-----------------------------------------------------------------------*/
/*  Definition related to Motor or PWM                                  */
/*-----------------------------------------------------------------------*/

/* motor spped control */
extern void MOTOR_SETSPD(byte _side, word Ratio);
```

3）【MOTOR_SETSPD 関数】：CN6 及び CN11をPWM信号で使用する際使用可能です。

MOTOR_SETSPD(byte {PWM の選択 }, word { 速度 });

●第一引数で PWM の選択を行います。

「1」: CN11 の PWM が選択されます。　PWM_SIDE_CN11

「0」: CN6 の PWM が選択されます。　PWM_SIDE_CN6

●第二引数でデューティー比を設定します。

0x0000 ～ 0xfffe までの間の値を設定して下さい。

```
/* motor speed rank setting */
extern void MOTOR_SPD(byte _side, byte _step);
```

4）【MOTOR_SPD 関数】：CN6 及び CN11をPWM信号で使用する際使用可能です。（PWM の設定を段階で行います。）

MOTOR_SPD(byte {M1,M2 の選択 },byte{ 速度段階 });

●第一引数で PWM の選択を行います。

「1」: CN11 の PWM が選択されます。　PWM_SIDE_CN11

「0」: CN6 の PWM が選択されます。　PWM_SIDE_CN6

●第二引数で速度の段階を設定します。

「0」: 高速　MOTOR_STEP_MAX

「1」: 中速　MOTOR_STEP_MID2

「2」: 低速　MOTOR_STEP_MID1

「3」: 超低速　MOTOR_STEP_MIN

「4」: 速度0　MOTOR_STEP_ZERO

```
/* motor stop */
extern void MOTOR_STOP(void);
```

5）【MOTOR_STOP 関数】：CN7～CN10、CN11（または、CN11とCN6）をモータドライバ制御へ使用した場合使用可能です。

例）MOTOR_STOP(void);

```
#define PWM_SIDE_CN11    1        // motor M2 side (M1 side)
#define PWM_SIDE_CN6     0        // motor M1 side (M1 only)

#define MOTOR_STEP_MAX   0        // Motor Speed max value
#define MOTOR_STEP_MID2  1        // Motor Speed middle2 value
#define MOTOR_STEP_MID1  2        // Motor Speed middle1 value
#define MOTOR_STEP_MIN   3        // Motor Speed mini value
#define MOTOR_STEP_ZERO  4        // Motor Speed zero value
```

・第一引数：PWMの選択を行います。

・第二引数：速度の段階を設定します。

【MOTOR_SETSPD 関数 記述例】

MOTOR_SETSPD (PWM_SIDE_CN11 , 0xfffe); または、

MOTOR_SETSPD (1 , 0xfffe);

【MOTOR_SPD 関数 記述例】

MOTOR_SPD (PWM_SIDE_CN11 , MOTOR_STEP_MID2); または、

MOTOR_SPD (1 , 1);

【MOTOR_STEP_ZERO 使用例】

●CN11からPWM信号を出力させない場合

MOTOR_SPD (PWM_SIDE_CN11 , MOTOR_STEP_ZERO); または、

MOTOR_SPD (1 , 4);

●CN6からPWM信号を出力させない場合

MOTOR_SPD (PWM_SIDE_CN6 , MOTOR_STEP_ZERO); または、

MOTOR_SPD (0 , 4);

```
#define M3           CN5          // 3th motor
```

6）【M3 ドライバ制御】：CN5をDigital in / outで使用する際使用可能です。
M3 と CN5 は同じです。

M3 = 1 ； と CN5 = 1 ； は同じです。

```
#define M1_IN1          CN7             // M1 motor mode select in1-pin
#define M1_IN2          CN8             // M1 motor mode select in2-pin
#define M2_IN1          CN9             // M2 motor mode select in1-pin
#define M2_IN2          CN10            // M2 motor mode select in2-pin
```

7)【モータの回転方向設定】: モータドライバを制御する際に使用可能です。

［CN7］と［M1_IN1］、［CN8］と［M1_IN2］、
［CN9］と［M2_IN1］、［CN10］と［M2_IN2］は同じです。

```
/*----------------------------------------------------------------------*/
/*  Definition related to AD Converter                                  */
/*----------------------------------------------------------------------*/

/* AD convert data get */
extern void AD_GETDATA(byte _channel, byte *pValue);

/* AD */
#define AD_CN3          4               // CN3  AD port channel
#define AD_CN4          5               // CN4  AD port channel
#define AD_CN6          0               // CN6  AD port channel
#define AD_CN11         1               // CN11 AD port channel
```

8)【 AD_GETDATA 関数】 : AD 変換の値を取得する関数です。

```
AD_GETDATA(byte _channel, byte *pValue);
```

● AD_GETDATA(byte {AD コネクタチャンネル} , byte {AD 値格納変数へのポインタ});

※ADコネクタチャンネルは左記の4つありますが CN6 と CN11 の使用については注意が必要です。

```
/*----------------------------------------------------------------------*/
/*  Definition related to I2C                                           */
/*----------------------------------------------------------------------*/

/* I2C data get */
extern void I2C_READ( byte I2C_device, word Address, byte *Data);
```

9）【I2C_READ 関数】 ：I2Cメモリのアドレスを指定して値を読み込む関数です。

I2C_READ(byte I2CAddress, word Address, byte *Data);

I2C_READ(byte {I2Cデバイス番号}, word {アドレス}, byte {値格納変数へのポインタ});

例　I2C_READ（I2C_MEM_ADD, 0x0100, &value);

●第一引数　：I2Cデバイスの番号
MPUボードのI2Cメモリは「I2C_MEM_ADD」を指定して下さい。

●第二引数　：I2Cメモリのアドレス 0 ～ 0xffff(64Kbyte）の値が設定できます。
MPUボードのI2Cメモリの制限によってアクセスできる範囲が異なります。

●第三引数　：取得した値を格納する変数へのポインタを指定します。

```
/* I2C data write */
extern void I2C_WRITE( byte I2C_device, word Address, byte Data);
```

10）【I2C_WRITE 関数】 ：I2Cメモリのアドレスを指定して値を書き込む関数です。

I2C_WRITE(byte I2CAddress, word Address, byte Data);

I2C_WRITE(byte {I2C デバイス番号 }, word { アドレス }, byte { 値格納変数 });

例　I2C_WRITE（I2C_MEM_ADD, 0x0100, value);

●第一引数　：I2Cデバイスの番号
MPUボードのI2Cメモリは「I2C_MEM_ADD」を指定して下さい。

●第二引数　：I2Cメモリのアドレス 0 ～ 0xffff(64K）の値が設定できますが、MPUボードのI2Cメモリの制限によってアクセスできる範囲が異なります。

●第三引数　：格納する値を保持している変数を指定します。

```
/* I2C */
#define I2C_DEVICE1     0x50 << 1       // I2C memory device address
#define I2C_DEVICE2     0x51 << 1       // I2C memory device address(24C1024)
```

11)【I2C デバイス番号】：基板上にあるメモリの I2Cデバイス番号です。

```
I2C_DEVICE1          0x50 << 1       // I2C (memory slave) address
```

●I2C＿READ関数やI2C＿WRITE関数の第一引数に使用します。
※この関数では64kbyteまでしかサポートできないため、24C1024の場合残りの
64kbyte のサポートは下記の関数を使用します。

```
I2C_DEVICE2         0x51 << 1 // I2C (memory slave) address(24C1024)
```

```
/*-------------------------------------------------------------------------*/
/*  Definition related to the Communication                                */
/*-------------------------------------------------------------------------*/

/* SCI Data Send */
extern void DATA_SEND(byte sendData);
```

12)【DATA_SEND 関数】： MPUボードから「通信ボード」経由で PC や他の機器とRS－232C接続でデータを送ります。
※ HW 設計画面でCN6へRS232Cタイルを配置しておく必要があります。

```
DATA_SEND(byte sendData);
```

```
/* SCI Data Recive */
extern byte DATA_RECIVE(void);
```

13)【DATA_RECIVE 関数】：PCや他の機器からRS－232C接続でデータを受け取ります。
※ HW 設計画面で CN6 へ RS232Cタイル を配置しておく必要があります。

```
DATA_RECIVE(byte reciveData);
```

```
/*-------------------------------------------------------------------------*/
/*  Definition related to the Timer delay                                  */
/*-------------------------------------------------------------------------*/

/* 10msec  delay */
extern void DELAY_SEC(byte _time);
```

14)【DELAY_SEC 関数】：ディレイ関数。指定した時間待機します。

```
DELAY_SEC(byte _time);
```

●第一引数で待機する秒数を設定します。
「1」で 0.01 秒 (10msec) です。0.1 秒の場合は「10」となります。

例） DELAY_SEC(0x10);

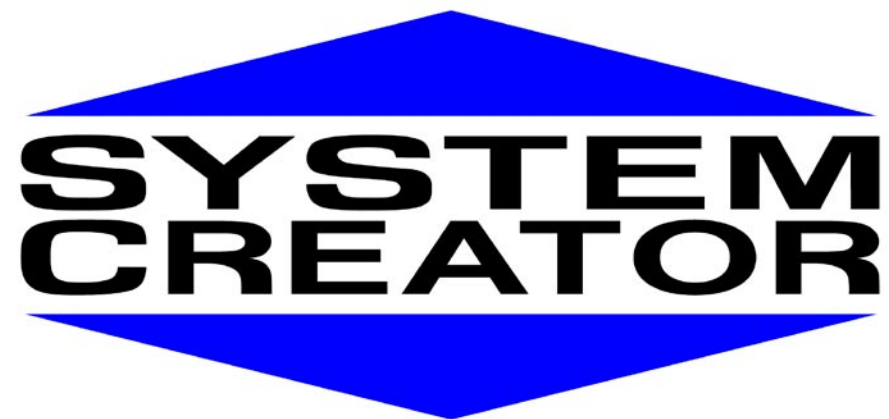
```
/*  5msec  delay */
extern void fineDELAY_SEC(word _time);
```

15)【fineDELAY_SEC 関数】：ディレイ関数。指定した時間待機します。

```
fineDELAY_SEC(word _time);
```

●第一引数（word 型 0 ～ 0xffff の範囲）で秒数を設定します。
「1」で 0.005 秒（5msec）です。

```
/*------------------------------------------------------------------------*/
/*  Definition related to the Initialization                              */
/*------------------------------------------------------------------------*/

/* interrupt initialize */
extern void INTERRUPT_INIT(void);

/* Port io initialize */
extern void PORT_INIT(void);

/* hardware setting */
extern void TICOLA_INIT(void);

/* PWM1 initialize */
extern void PWM1_INIT(void);

/* PWM2 initialize */
extern void PWM2_INIT(void);

/* ADC initialize */
extern void ADC_INIT(void);

/* SCI initialize */
extern void SCI_INIT(void);

/* I2C initialize */
extern void I2C_INIT(void);
```

16)【イニシャライズ関連関数】: HW 設計画面と連動して自動的に処理されます。

サンプルプログラム

3. サンプルプログラム

サンプルプログラムには下記の入出力基板を使用して作成しています。

●JES－7002　　I/O Connecter ボード
●RDI－205A　　4Bit ディジタルスイッチボード
●RDI－207　　アナログ実験ボード

3-1. サンプルプログラムの開き方

ツールバー「File」→「Sample Program Open」から任意のプロジェクトを選択します。

●⚠注 意●

①アプリケーション上で開かれたサンプルプログラムデータの直接の編集はできません。
サンプルプログラムを編集する場合は、［マイドキュメント］や任意の［ローカルディスク（C: や D:）］へ［別名で保存］し、そのデータをアプリケーション上で開いて編集して下さい。

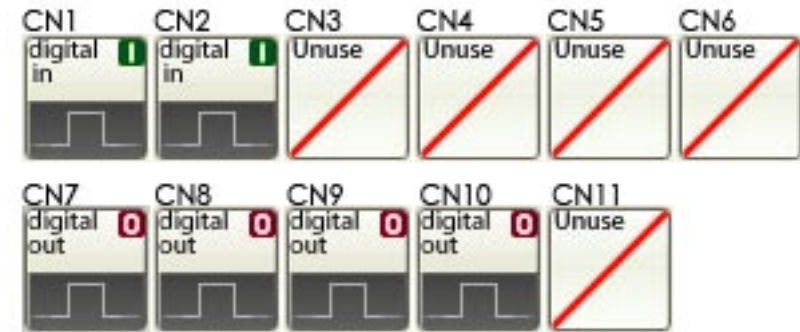
②インストール後全てのユーザー（管理者及びゲストアカウント）で使用可能です。

③ゲストアカウントでSC7000のサンプルプログラムを使用する場合、初期設定でサンプルプログラムのコピーを作成する必要があります。
作成手順は以下の通りです。
ツールバー［File］→［Sample Program Open］→［Sample restoration］をクリックします。以上でコピーの完了です。

④OS［Vista］のバージョン（エディション）によっては、サンプルプログラムの編集が可能となる場合があります。編集されたサンプルプログラムがサンプルとして機能しない場合は、［Sample restoration］をクリックすることでサンプルプログラムの復元が可能です。

3-2. ディジタルInput / Output　A_D_inout

■ HW 設計

［Preference Mode］：Standard モード

■ main.c

```
/*********************************** 中　略 ***********************************/

#include "ES7000.h"         // Hardware definition
#include "SystemCreator.h"  // SystemCreator Hardware definition
#include "ES7000Lib.h"      // Library definition
#include "vectors.h"        // Interruption vector definition
#include "initialize.h"     // Ininitalize
#include "interrupt.h"      // Interruption

/*----------------------------------------------------------------------------*/
/* Variable declaration                                                       */
/*----------------------------------------------------------------------------*/

/*----------------------------------------------------------------------------*/
/* Prototype declaration                                                      */
/*----------------------------------------------------------------------------*/

void Dout0( void );                                          // 出力０
void Dout1( void );                                          // 出力１
void Dout2( void );                                          // 出力２
void Dout3( void );                                          // 出力３

/*----------------------------------------------------------------------------*/
/*  main                                                                      */
/*----------------------------------------------------------------------------*/

void main(void) {
```

```
        init();              //Initialization

        while(1){

                if ( CN1 == 0 ){                                  //  ＣＮ１がＯＦＦなら
                        if ( CN2 == 0 ){
                                Dout0;                            //  CN1 = 0 ・ CN2 = 0
                                DELAY_SEC(50);                    //  10mSEC タイマ × 50
                        }else{
                                Dout1;                            //  CN1 = 1 ・ CN2 = 0
                                DELAY_SEC(50);                    //  10mSEC タイマ × 50
                        }
                }else{                                            //  ＣＮ１がＯＮなら
                        if ( CN2 == 0 ){
                                Dout2;                            //  CN1 = 1 ・ CN2 = 0
                                DELAY_SEC(50);                    //  10mSEC タイマ × 50
                        }else{
                                Dout3;                            //  CN1 = 1 ・ CN2 = 1
                                DELAY_SEC(50);                    //  10mSEC タイマ × 50
                        }
                }

                COPCTL = 0;
        }
}

/*----------------------------------------------------------------------------*/
/*  function                                                                  */
/*----------------------------------------------------------------------------*/

// 出力０
void Dout0( void ){
        CN7  = 1;
        CN8  = 0;
        CN9  = 0;
        CN10 = 0;
}

// 出力１
void Dout1( void ){
        CN7  = 0;
        CN8  = 1;
        CN9  = 0;
        CN10 = 0;
}

// 出力２
void Dout2( void ){
        CN7  = 0;
        CN8  = 0;
        CN9  = 1;
        CN10 = 0;
}

// 出力３
void Dout3( void ){
        CN7  = 0;
        CN8  = 0;
        CN9  = 0;
        CN10 = 1;
}
```

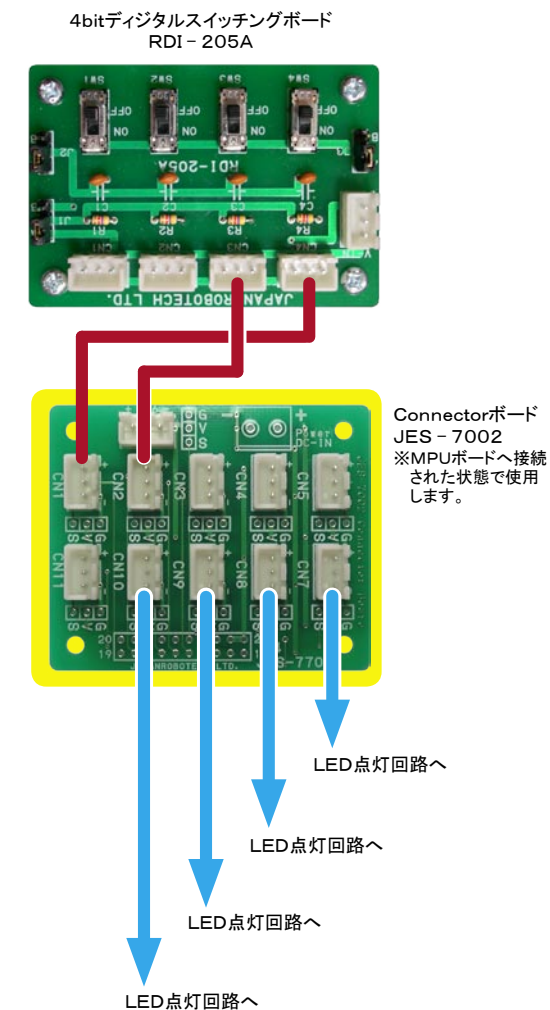
■プログラム解説

ポートCN１及びCN2の状態により［LED1～4］を点・消灯させるプログラム例です。

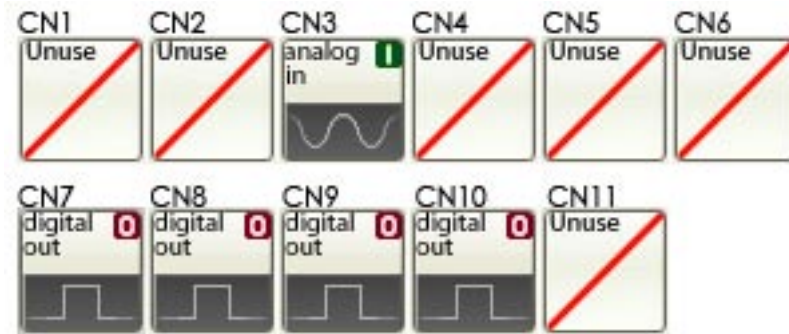
●JES－7002 ポートの設定

CN1＝ Digital In (SW In)
CN2＝ Digital In (SW In)
CN7＝ Digital Out (LED Out)
CN8＝ Digital Out (LED Out)
CN9＝ Digital Out (LED Out)
CN 10 ＝ Digital Out (LED Out)

if 文にて CN1及びCN2 の4つの状態を判断し、Functionで定義している状態でLEDを点灯させます。

■入出力機器セットアップ例

3-3. アナログ Input/ ディジタル Output B_A_in

■ HW 設計

［Preference Mode］：Standard モード

■ main.c

```
/************************************ 中　略 ************************************/

/*----------------------------------------------------------------------------*/
/* Variable declaration                                                       */
/*----------------------------------------------------------------------------*/

byte Value;                                         // アナログ値用変数

/*----------------------------------------------------------------------------*/
/* Prototype declaration                                                      */
/*----------------------------------------------------------------------------*/

/*----------------------------------------------------------------------------*/
/*  main                                                                      */
/*----------------------------------------------------------------------------*/

void main(void) {

        init();             //Initialization

        while(1){
                AD_GETDATA(AD_CN3,  &Value);  // CN3 のアナログ値を取得

                Value = Value >> 4;           // 取得したアナログ値上位 4bit を下位 4bit へ移動
                LED_OUT( Value );             // 移動した値を CN7 ～ CN10 へ出力

                COPCTL = 0;
        }
}
```

■プログラム解説

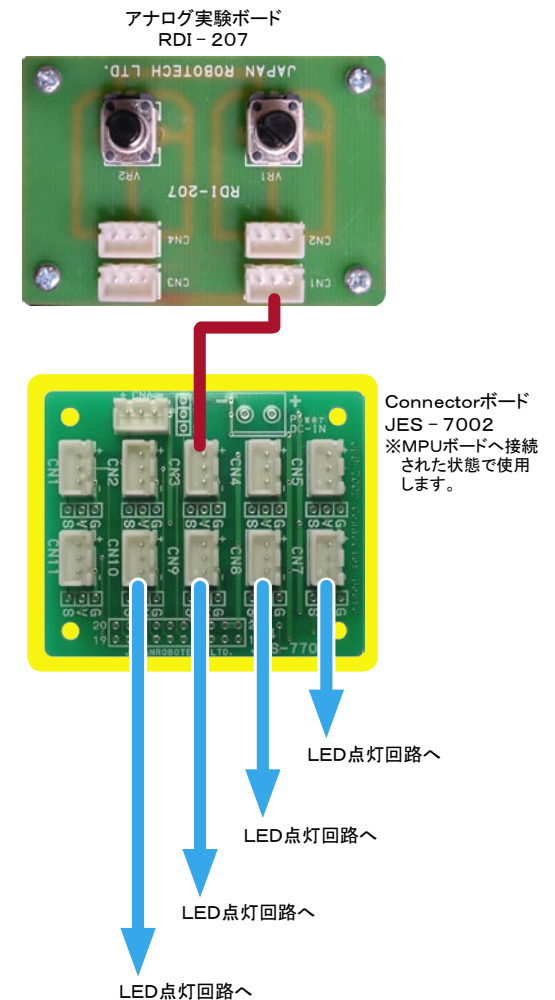
CN3 のアナログ値を取得し、4Bit LED を点灯させるプログラム例

●JES－7002 ポートの設定
- CN3＝ Analog In
- CN7＝ Digital Out（LED Out）
- CN8＝ Digital Out（LED Out）
- CN9＝ Digital Out（LED Out）
- CN 10 ＝ Digital Out（LED Out）

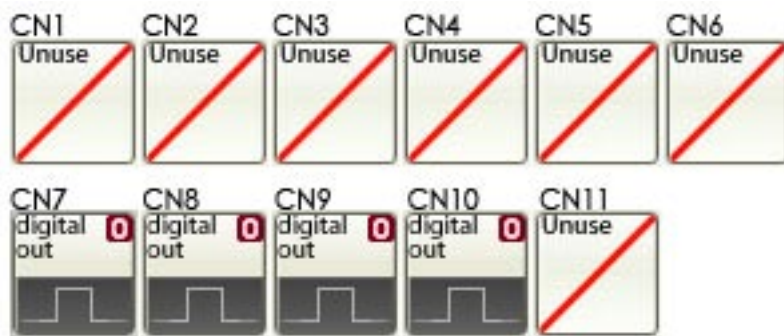
AD_GETDATA で CN3 のアナログ値を取得し、4ビットシフトした後 LED_OUT で点灯させます。

■入出力機器セットアップ例

アナログ実験ボード
RDI－207

Connectorボード
JES－7002
※MPUボードへ接続された状態で使用します。

3-4. SW2による割り込み C_Int_IRQ

■ HW 設計

［Preference Mode］：Advanceモード

■ main.c

```
/******************************************************************************
 * Program Name: C_Int_IRQ
 * Author      : JAPANROBOTECH - Y
 * Date        : 2007-10-05 13:18:58
 * explanation : スイッチ割り込み（ＳＷ２・ＩＲＱ）プログラム例
 *
 *                     JAPANROBOTECH C-Language Development Environment
 ******************************************************************************/

#include "ES7000.h"          // Hardware definition
#include "SystemCreator.h"   // SystemCreator Hardware definition
#include "ES7000Lib.h"       // Library definition
#include "vectors.h"         // Interruption vector definition
#include "initialize.h"      // Ininitalize
#include "interrupt.h"       // Interruption


/*----------------------------------------------------------------------------*/
/* Variable declaration                                                       */
/*----------------------------------------------------------------------------*/
byte i;                                          //タイムディレー用変数
byte led_value;                                  //ＬＥＤ点灯用変数
extern byte int_sw;                              //割り込み状態変数

/*----------------------------------------------------------------------------*/
/* Prototype declaration                                                      */
/*----------------------------------------------------------------------------*/

/*----------------------------------------------------------------------------*/
/*  main                                                                      */
/*----------------------------------------------------------------------------*/

void main(void) {

        init();              //Initialization

        for (i=0; i<5; ++i){                     //タイムディレー用点滅処理
                LED_OUT( 0x0F );
                DELAY_SEC(20);
                LED_OUT( 0x00 );
                DELAY_SEC(20);
        }

        int_sw = 0;                              //割り込みによる状態変数クリア
        INTERRUPT_INIT();                        //割り込み初期設定
        led_value = 0;                           //ＬＥＤ表示用変数クリア

        while(1){
                if ( int_sw == 0 ){
                LED_OUT( led_value );            //ＬＥＤ表示処理
                DELAY_SEC( 20 );                 //タイムディレー
                ++led_value;                     //ＬＥＤ表示データアップ処理
                        if (led_value == 0x10){  //ＬＥＤ表示上限値判断処理
                                led_value = 0;   //ＬＥＤ表示クリア処理
                        }
                }
                COPCTL = 0;
        }
}
```

■ interrupt.c

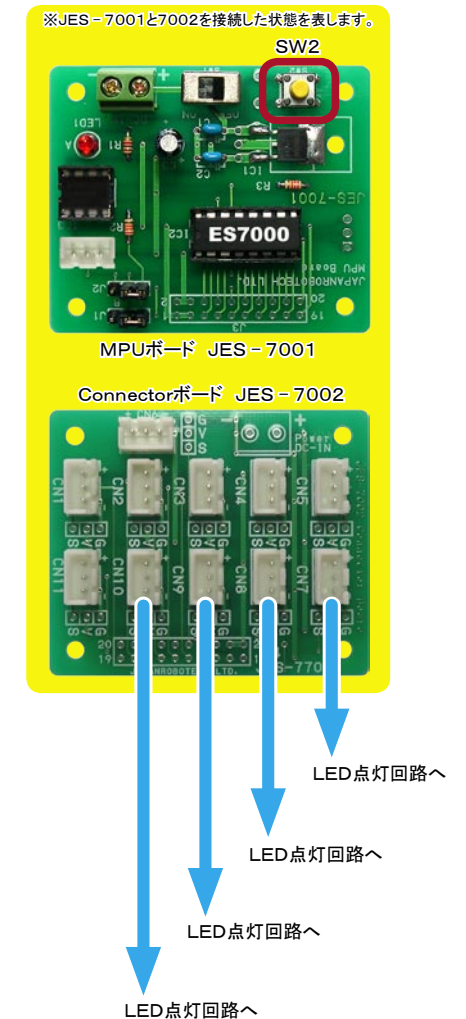
```
/*************************************** 中　略 ***************************************/
#include "ES7000.h"          // Hardware definition
#include "SystemCreator.h"   // SystemCreator Hardware definition
#include "ES7000Lib.h"       // Library definition


/*----------------------------------------------------------------------------*/
/* Variable declaration                                                       */
/*----------------------------------------------------------------------------*/

byte int_sw;                                              //割り込み状態変数

/*************************************** 中　略 ***************************************/

/*----------------------------------------------------------------------------*/
/* IRQ interrupt                                                              */
/*----------------------------------------------------------------------------*/

//#pragma TRAP_PROC
void _IRQ_Interrupt(void) {
        __DI();

        if ( int_sw == 0){
                DELAY_SEC( 30 );                    //タイムディレー
                int_sw = 1;                         //停止値セット処理
        }else{
                DELAY_SEC( 30 );                    //タイムディレー
                int_sw = 0;                         //停止値クリア処理
        }

        __EI();
}
```

■プログラム解説

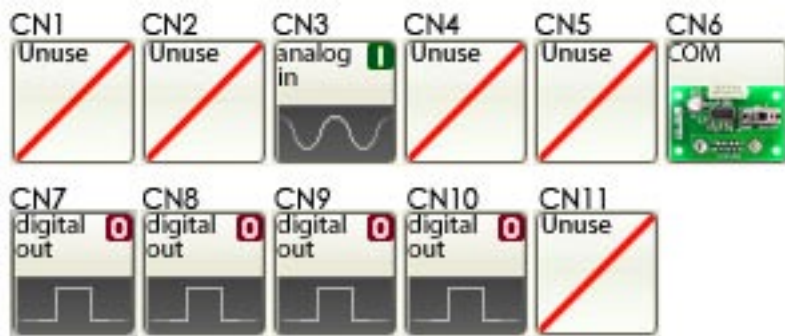
MPUボード JES－7001基板上にある SW2（IRQ：赤枠）スイッチを使用して、SW2を押すと、LEDが順次点灯→停止を繰り返すプログラム例です。

●JES－7002 ポートの設定
- CN7＝ Digital Out (LED Out)
- CN8＝ Digital Out (LED Out)
- CN9＝ Digital Out (LED Out)
- CN 10 ＝ Digital Out (LED Out)
- SW2＝ IRQ（7001基板上）

interrupt.cシートのIRQ interrupt領域で int_sw の値を変更し、int_sw 値の状態によりLEDを点灯を行います。

■入出力機器セットアップ例

3-5. データロガー D_Log

■ HW 設計

［Preference Mode］：Advanceモード

■ main.c

```
/*****************************************************************************
 *  Program Name:  D_Log
 *  Author       : JAPANROBOTECH - Y
 *  Date         : 2007-10-16 15:02:46
 *  explanation : データロガー（通信）サンプルプログラム例
 *
 *                        JAPANROBOTECH C-Language Development Environment
 *****************************************************************************/

#include "ES7000.h"          // Hardware definition
#include "SystemCreator.h"   // SystemCreator Hardware definition
#include "ES7000Lib.h"       // Library definition
#include "vectors.h"         // Interruption vector definition
#include "initialize.h"      // Ininitalize
#include "interrupt.h"       // Interruption
#include "sub_Log.h"         // サブルーチン

/*---------------------------------------------------------------------------*/
/* Variable declaration                                                      */
/*---------------------------------------------------------------------------*/
byte i;                                          //タイムディレー用変数

/*---------------------------------------------------------------------------*/
/* Prototype declaration                                                     */
/*---------------------------------------------------------------------------*/



/*---------------------------------------------------------------------------*/
/*  main                                                                     */
/*---------------------------------------------------------------------------*/

void main(void) {

        init();            //Initialization

        for (i=0; i<20; ++i){                    //タイムディレー用点滅処理
                LED_OUT( 0x0F );                 //J1 = B・J2 = A の状態で電源 ON
                DELAY_SEC(20);                   //(←この処理間に)J2 を A から B へ切替
                LED_OUT( 0x00 );
                DELAY_SEC(20);
        }

        while(1){
                log_test();                      // <-サブルーチンをコール

                COPCTL = 0;
        }
}
```

■ sub_Log.c

```
/*****************************************************************************
 *  Program Name:
 *  Author       :
 *  Date         : 2007-10-16 15:05:26
 *  explanation :
 *
 *                                  JAPANROBOTECH C-Language Development Environment
 *****************************************************************************/

#include "ES7000.h"             // Hardware definition
#include "SystemCreator.h"      // SystemCreator Hardware definition
#include "ES7000Lib.h"          // Library definition

/*---------------------------------------------------------------------------*/
/*  AD Data to Logger (LED Flash)                                            */
/*---------------------------------------------------------------------------*/
void log_test(){
        byte Value;                             // <-- ローカル変数も OK
                                                //(スタックのサイズの関係であまり多くは使用できない)

        while(1){
                AD_GETDATA(AD_CN3,  &Value);    // CN3 のアナログ値を取得

                DATA_SEND(Value);               //取得した光センサの値を RS232C へ出力
                Value = Value >> 4;             //取得した光センサの値上位 4bit を下位 4bit へ移動
                LED_OUT( Value );               //取得した光センサの値 (元上位 4bit) を出力
                DELAY_SEC(10);                  //10msec × 10 = 100msec タイムディレー
                COPCTL = 0;
        }
}
```

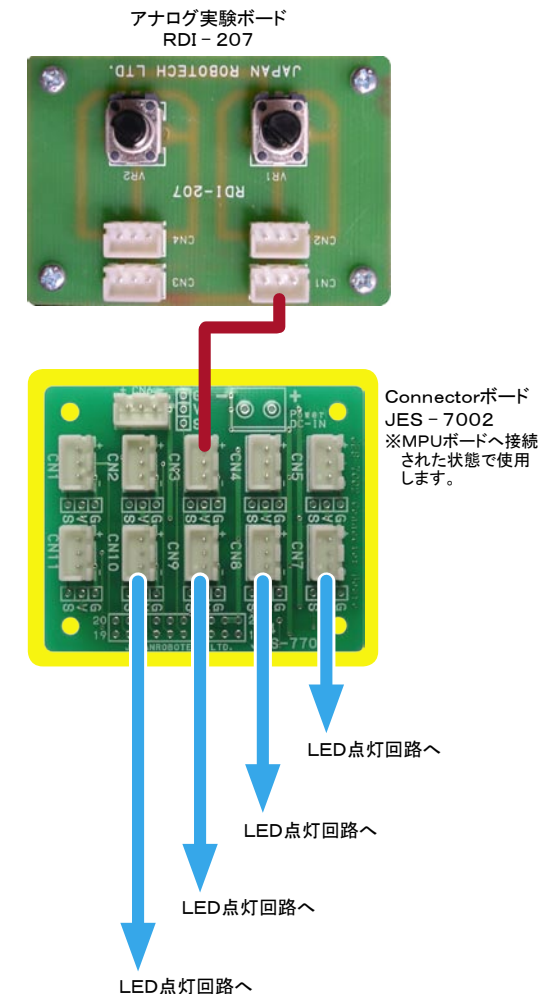
■プログラム解説

CN3 のアナログ値をCOMポートへ出力し、上位4bit値でCN7 ～CN 10の LED を点灯するプログラム例です。

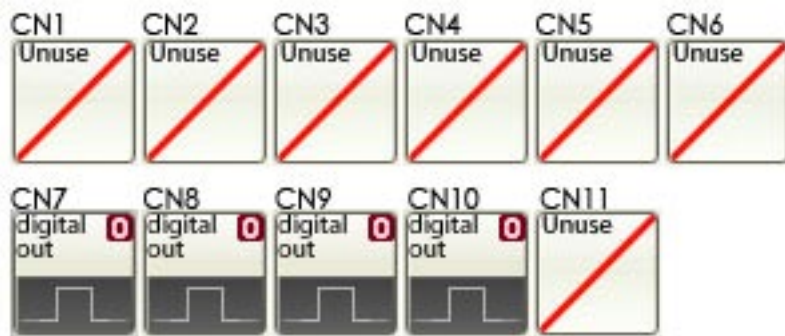
●JES－7002 ポートの設定

- CN3＝ Analog In
- CN6＝ COM (RS-232C)
- CN7＝ Digital Out (LED Out)
- CN8＝ Digital Out (LED Out)
- CN9＝ Digital Out (LED Out)
- CN 10 ＝ Digital Out (LED Out)

サブルーチンを使用してCN3のアナログ値取得後COMポートへの出力及び上位4Bit 値を LED へ出力します。
実行についてはmain for文動作中（LED点滅処理）に基板上 J2の切替（通信回路接続）を行うことが重要です。

■入出力機器セットアップ例

3- 6. I2Cメモリ使用例　E _ I2C

■ HW 設計

［Preference Mode］：Advanceモード

■ main.c

```
/*******************************************************************************
 *  Program Name: E_I2C
 *  Author       : JAPANROBOTECH - Y
 *  Date         : 2007-11-07 16:09:19
 *  explanation : I2C メモリ使用例
 *
 *                                  JAPANROBOTECH C-Language Development Environment
 *******************************************************************************/

#include "ES7000.h"          // Hardware definition
#include "SystemCreator.h"   // SystemCreator Hardware definition
#include "ES7000Lib.h"       // Library definition
#include "vectors.h"         // Interruption vector definition
#include "initialize.h"      // Ininitalize
#include "interrupt.h"       // Interruption


/*---------------------------------------------------------------------------*/
/* Variable declaration                                                      */
/*---------------------------------------------------------------------------*/
byte i2c_data;
word address;

/*---------------------------------------------------------------------------*/
/* Prototype declaration                                                     */
/*---------------------------------------------------------------------------*/



/*---------------------------------------------------------------------------*/
/*  main                                                                     */
/*---------------------------------------------------------------------------*/

void main(void) {

         init();             // Initialization

         while(1){
                  address = 0;                              // 初期設定（0にセット）
                  i2c_data = 0;                             // 初期設定（0にセット）
                  LED_OUT( 0 );                             // ＬＥＤ出力クリア

                  for (address=0; address<255; ++address){  // address 0～255 回設定
                           ++i2c_data;                      // i2c_data 1加算
                           I2C_WRITE( I2C_DEVICE1, address, i2c_data);  // I2C メモリへ
                                                                        // データ書き込み
                           COPCTL = 0;
                  }

                  for (address=0; address<255; ++address){  // address 0～255 回設定
                           I2C_READ( I2C_DEVICE1, address, &i2c_data);  //I2C メモリへ
                                                                        // データ読み込み
                           LED_OUT( i2c_data );             // 読み込みデータをＬＥＤ
                                                            // へ出力
                           DELAY_SEC(5);                    // 10mSEC タイマ × 5
                           COPCTL = 0;
                  }

                  COPCTL = 0;
         }
}
```

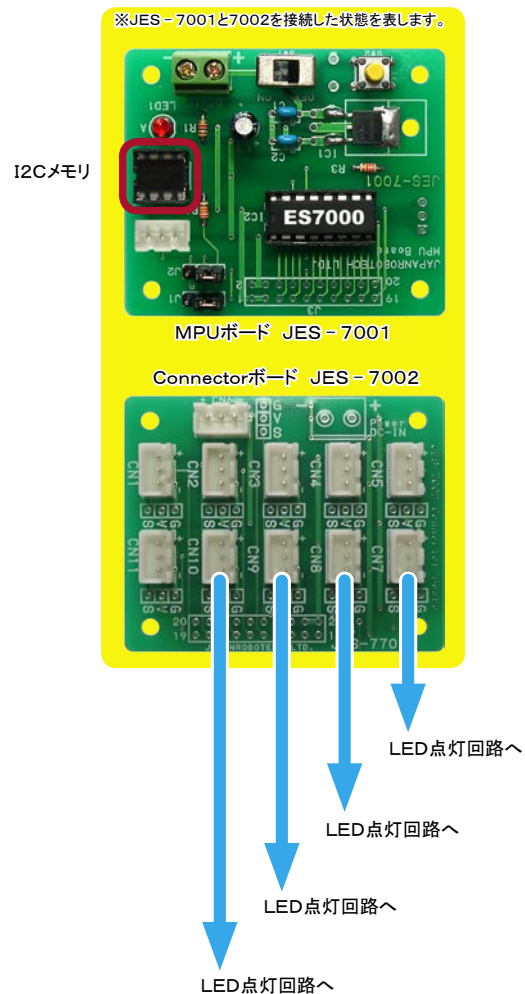
■プログラム解説

MPUボード JES－7001搭載 I2Cメモリへ 0 ～ 255 の値を書込んだ後、データを読出し 、CN7～CN10 の LED を点・消灯するプログラム例です。

●JES－7002 ポートの設定
CN7＝ Digital Out (LED Out)
CN8＝ Digital Out (LED Out)
CN9＝ Digital Out (LED Out)
CN10 ＝Digital Out (LED Out)

main にある最初の for 文にて書き込み処理を行う、次の for 文にて読出し, LED 点灯処理を行います。

■入出力機器セットアップ例

I2Cメモリ

サポート情報

ジャパンロボテック カスタマーセンター

メール　info@japan - robotech . com
電話　092- 821- 4421（受付時間 AM10：00～PM17：00）
ファックス　092- 821- 4422

第1版（1,00）平成19年11月20日
第2版（1.01）平成20年 1月25日

発行者 河 野 孝 治

発行所
株式会社 JAPAN ROBOTECH
〒812-0025 福岡市博多区店屋町 4 番 18 号 冷泉ビル 33 号
R&D CENTER
〒814- 0001 福岡市早良区百道浜2－3－2 TNC放送会館2F ロボスクエア内
http://www.japan-robotech.com/systemcreator/index.html
http://www.japan-robotech.com/